PC-9800シリーズ

ザ・プロテクト

井上智博



SHUWA SYSTEM TRADING CO.,LTD.

PC-9800 SERIES

THE PROTECT

SHUWA SYSTEM TRADING CO.,LTD.

まえがき

「プロテクトってなあに」
本書を手にしているあなたは、もちろん知っていますね。
そうです。ソフトを買うとマニュアルに書いてあるこれです。

『このフロッピディスクはバックアップをとることができません。』
探求心の強いあなたはそれにもめげずバックアップを試みたことでしょう。
でも、やはりバックアップはとれません。
　あるソフトではエラーが出て止まってしまうでしょう。また、あるソフトではコピーは正常終了したにも拘らず、実行させると動かないといった症状かもしれません。

ではどうしてバックアップがとれないのでしょうか。そのなぞ解きがこの本です。
もちろん、貧弱なプロテクトは少し工夫をすればバックアップをとることができます。しかし、現在の技術ではどうがんばってもコピーできないプロテクトも存在します。
「えー」と驚く人もいるでしょう。
こう反論する人もいるかも知れません。
「ＰＣ－９８００を使って作ったフロッピディスクを同じ装置を使って作れないはずはない」と。

　この本を読み終わった時にもう一度、この反論が言えるでしょうか。

井上　智博

本書の構成

本書は知識編および応用編からなっています。

知識編ではプロテクトの作成、解析に必要なあらゆる知識を体系的に網羅しています。自分はプロテクトに関してかなり知っているとおっしゃる読者の方も、知識編から注意深く読まれることをお勧めします。読み進むに従ってきっと新しいプロテクトのアイデアを思いつくことでしょう。

応用編では実際のプロテクト例を扱います。知識編で解説した知識をどのように用いればプロテクトになるかを読み取ってください。すべての応用例には、その作成法を示し、多くのプログラム例も掲載しました。読者の皆さんはこれらのプロテクトを自分で作って試してみることができます。また、コピー可能なプロテクトに関しては、コピー方法をも解説してあります。ぜひ御活用ください。プロテクトはＢＡＳＩＣの範囲内では実現できないものが多く、本書に掲載したプログラム例はアセンブラが主体となっていますが、プログラムには日本語で注釈を付してありますから、アセンブラが初めての方にも理解しやすいでしょう。

もう一つのプロテクト

いくら強いプロテクトでもプログラムを解読されてチェックルーチンをはずされてしまったのでは元も子もありません。そこで重要となるのがいかに解読しにくくするかです。究極のプロテクトが存在する現在、本当に求められているのは、この解読に対するプロテクトです。この解読プロテクトに関しては次の機会に紹介したいと思います。

目　次

◎

知識編

1　プロテクトの原理

1．1　プロテクトとは何だろう　3
プロテクトの誕生　3
コピーできないとは・・・　3
どうしてコピーできないか？　4
OSのフォーマットに従わなくても
そのOS上で動くのか？　5
コピーツールの出現　6
プロテクトの限界は？　6
1．2　究極のプロテクトは可能か？　7
プロテクトが成り立つための条件とは・・・　7
完全なコピーは不可能か　8
各トラックの完全なコピーは可能か　12
くせもののFDC　18

◎

2　FDCの知識

2．1　FDCの想定するフォーマット　20
各部の説明　22
2．2　FDCのコマンドを検討する　28

マルチセクタとＴＣとマルチトラック　29
各パラメータの意味　30
Read Diagnostic　33
Write ID　37
Read ID　38
Write Data, Write Deleted Data　39
Read Data　41
2．3　読めない部分があるってホント？　43
ＩＤがあるのにアンフォーマット　44
プロテクトへの可能性　45
2．4　ハングアップするってホント？　46
ハングアップの条件　46
どうやって条件を満たすか　47
何度かリトライする　49
プロテクトへの可能性　49
2．5　データビットとクロックビット　51
ＦＭとＭＦＭ　51
単密度と倍密度　53
単密度を倍密度で読むと　54
ミッシングクロック　55

◎

3　ＢＩＯＳの知識

3．1　ＢＩＯＳとは何だろう　57
コマンド一覧　57
3．2　ＢＩＯＳにできないこと　60
3．3　こうすればＢＩＯＳでも使える　62

3．4　ＢＩＯＳの基本的な使い方　65
ＢＩＯＳのコール手順　65
入 力　66
出 力　72
リターンコード　72

◎

4　プロテクトの知識

4．1　プロテクトをかけるには　73
特殊なハードウエアで作成した場合　73
手順がわからない場合　74
誤差のために同じものができない場合　74
4．2　作成手順をわからなくするには　75
コマンドの順番　75
タイミング　75
パラメータ　76
データ　77
4．3　ビットずれのいろいろ　78
安定したビットずれ　79
不安定なビットずれ　80
4．4　回転数とトラック容量　83
4．5　リセットを使ってみると　85
4．6　プロテクト組み込み上の注意　86
Ｎ88－ＢＡＳＩＣの場合　86
ＭＳ－ＤＯＳの場合　86
ＣＰ／Ｍ－８６の場合　87

応用編

1　基本テクニック

1．1　標準以外のセクタ長を使う　95
1．2　単密度を使う　100
1．3　ディリーテッドレコードを使う　104
1．4　セクタの順番をチェック　107
1．5　セクタの位置をチェック　112
1．6　ＩＤのＣが実際と異なる　117
1．7　ＩＤのＨが実際と異なる　121
1．8　ＩＤのＮが実際と異なる　124
1．9　標準より少ないセクタ数　127
1．10　標準より多いセクタ数　130
1．11　アンフォーマットを使う　134
1．12　オーバートラックを使う　136

◎

2　上級テクニック

2．1　１トラック中に異なるセクタ長が混在　139
2．2　１トラック中に同じＩＤが複数存在　148
2．3　ＧＡＰ３の長さをチェック　152
2．4　先頭セクタの位置をずらす　157
2．5　ＧＡＰ３にデータを書き込む　164
2．6　ＤＡＭを破壊する　166

◎

3　ビットずれの応用

3．1　データの書き込み有無をチェック　168
3．2　ずれの度合いをチェック　171
3．3　不安定度のチェック　174

◎

4　回転変動の応用

4．1　トラック容量を測定する　177
4．2　ＧＡＰ３の長さを変える　181
4．3　ＧＡＰ３にごみを入れる　184
4．4　普通では入らないセクタ数　186

◎

5　リセットの応用

5．1　データがあるのにＣＲＣエラー　188
5．2　ＩＤのＣＲＣエラー　191
5．3　ＤＡＭを破壊する　194
5．4　ＧＡＰ３にデータを書き込む　196
5．5　ＩＤだけのセクタ　199
5．6　単密度、倍密度の混在　201
5．7　アンフォーマットのデータを読む　203

◎

6　ハングアップの応用

6．1　コピーツールをハングアップ　206
6．2　ハングアップするかをチェック　209

⊙

付　録

1．　拡張ＢＩＯＳの製作　213
2．　トラック容量測定と調製ツール　233
3．　関連周辺ＬＳＩの使い方　242

PC-9800
SERIES
THE PROTECT

I　知識編

○プロテクトの原理
○FDCの知識
○BIOSの知識
○プロテクトの知識

1.1 プロテクトの原理 プロテクトとは何だろう

◎ プロテクトの誕生

　コンピュータには、そのハードウエアと同様にソフトウエアが必要なことは、御存知のことと思います。しかし一般には、コンピュータの価格をハードウエアだけでとらえることが多く、ソフトウエアが軽視されているような印象を受けます。それは、ハードウエアが形のある物体であるのに対して、ソフトウエアが目に見えない情報であることに原因があると思われます。近年、パーソナルコンピュータの普及とともにソフトウエアの需要が増大するにつれて、ソフトウエアをコピーして入手するという手段がとられ始めました。これは、ソフトウエアメーカーにとっては大きな損害となり、対抗手段として容易にはコピーできない工夫をソフトウエアに施すようになったのです。これをプロテクトと呼びます。

◎ コピーできないとは・・・

　当初、ユーザは、コピーするのにシステムに付属のユーティリティプログラム（N_{88}－BASICでは、"backup.n88"、MS－DOSでは、"DISKCOPY"など）を使いました。コピーできないとは、それらのプログラムでコピーできないということで、コピーしようとした場合の症状は次の２つに大別されます。

　①　コピープログラムが途中で異常終了する。

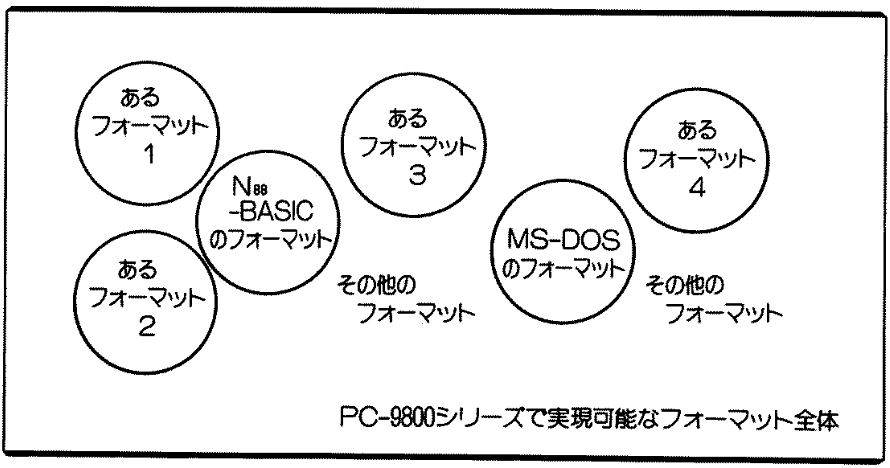
　②　コピープログラムは正常に終了するが、コピーしたものを実行させるとうまく動作しない。

◎どうしてコピーできないか？

まず、①の場合、これはコピープログラムの想像を超えるディスケットだったからなのです。N_{88}−BASICとMS−DOSではディスク上の物理的なフォーマットが違うように、数多くのフォーマットが作成可能です。ところが、"backup.n88"、"DISKCOPY"などは、そのOSで使われる特定のフォーマットしか想定していないため、それ以外のフォーマットが現われると異常終了してしまうのです。これは、そもそもPC−9800シリーズでは、いろいろなフォーマットを実現できるハードウエアを持ちながら、OSがそれらをサポートしていないため起こる現象です。

②の場合は、一応、正常に終了しますが、コピーされていない部分があったり、コピーしたつもりでも見方を変えると本物と違う部分があったりで、実行時にそれらをチェックしてひっかかるとまともに動作しないようにプログラムが組まれているのです。

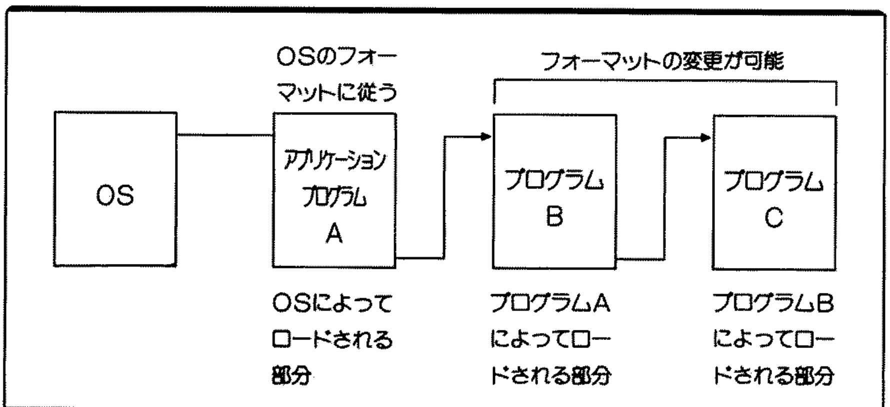
図1．1　ハードウエア自体はもっといろいろなフォーマットが可能



◎――――――ＯＳのフォーマットに従わなくても、そのＯＳ上で動くのか？

これは、初心者がよく抱く疑問です。また、「MS－DOS上で動くプログラムにプロテクトがかけられるはずはない」という人もいます。しかし、心配はいりません。ちゃんとプロテクトはかけられます。プログラムも動きます。N_{88}－BASIC、MS－DOS，CP／M－８６などのOSは、それぞれ特定のフォーマットを持ち、特定のファイル管理に従っています。しかし、フォーマットを変更した部分をOSを使って読むわけではないのです。OSが読み込むのはあくまでも正常な部分のみです。まず、OSを使ってアプリケーションプログラムの正常な部分を読み込み、次にそのプログラムを実行することによって、そこに組み込まれた専用の読み込みルーチンが働いて、フォーマットを変更した部分を読み込みます。さらに念を入れたものでは、新しく読み込んだ部分に次の読み込みルーチンがあってそこに制御を移すというものもあり、これはいくらでも繰り返せます。つまり、最初に読み込む部分だけOSに従っていればいいわけです。

図１．２　ＯＳに従うのは最初だけ



◎ コピーツールの出現

プロテクトの誕生に少し遅れて、プロテクトのかかったプログラムをコピーするプログラム、コピーツールが出現してきました。これは、"backup.n 88"などのように特定のフォーマットに限らず、トラック毎にどういうフォーマットかを自動的に調べ、同じフォーマットを実現することを目指したものです。コピーツールの出現によって、通常のプロテクトでは通用しなくなり、さらに複雑なプロテクトが生み出されてきました。

◎ プロテクトの限界は？

「いくらプロテクトを複雑にしても、本物のディスケットをPC－９８００シリーズで作成できるなら、それと同じ手順で作成すれば同じものができるはずだ」という意見もあります。これをもって「プロテクトには限界がある。必ずコピーできる」というならそれは間違いです。理由は２つです。

① もし、先の意見が正しいとしても、果たしてその手順がわかるかどうかという問題があります。プロテクトをかけた本人が教えてくれるなら別ですが、結果としてのディスクを見ただけで、作成方法がわかるかどうかは疑問です。

② 先の意見では見落としている点があります。それは誤差という問題です。例え同じ手順を用いても、誤差があれば同じものはできないのです。すべてのものには誤差はつきものです。問題はその誤差が検出できるほど大きいかどうか、もう一つは、誤差を問題にして本物まで動作しなくなることはないか、の２つでしょう。結論は、「心配ありません」。理由は読み進むにつれてわかってくるはずです。

1.2 プロテクトの原理 究極のプロテクトは可能か？

◎―――― プロテクトが成り立つための条件とは・・・

「絶対にコピーできないプロテクトは可能だろうか」。本節ではこの問題を基本に戻って別の観点から考えてみます。

プロテクトとは一般に次の状態が満たされることをいいます。

① 本物のディスケットでは正常に動作すること

② コピーされたディスケットでは正常に動作しないこと

これらを満たすためには本物と偽物（コピーされたディスケットのこと）を見分けることができなくてはなりません。どう見てもまったく同じものは見分けることができないわけで、プロテクトが成り立つためには両者の間になんらかの違いが生じる必要があります。また、その違いを調べることができなくてはなりません。

プロテクトが成り立つための必要条件

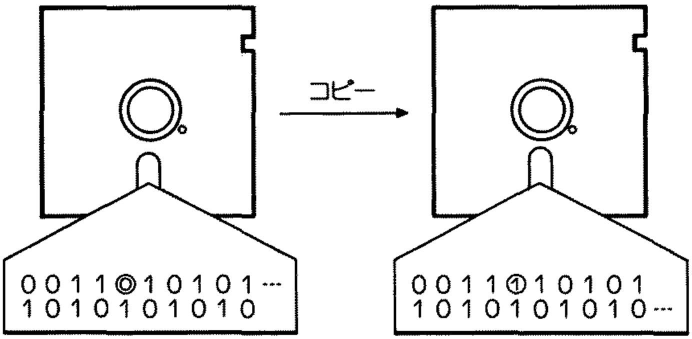
① 本物と偽物の間に違いが生ずること

② その違いをチェックできること

プロテクトにはこの２つを満たすことが必要で、つまり、たとえ本物と偽物に違いを生じさせることができたとしても、それをチェックすることができなければプロテクトにはならないのです。これは重要なことです。

では、実際にどのようにこれらを実現すればいいのでしょうか。

図1．3　違いが生ずる必要がある



プロテクトが成り立つためには必ず本物とコピーで違いが生じなくてはならない。

コピーする側に立って考えてみます。もし、完全に寸分の違いもなくコピーできたすると、どんなにチェックをしたって違いは発見できないのですからもうどうしようもありません。プロテクトはおしまいです。幸い、現在のところ完全に同じものを作ることはできないので、助かっているのです。そうなると問題はチェックの方法です。つまり、コピーする側は、そこで行なわれるチェックにひっかからない程度に同じものを作ろうとします。

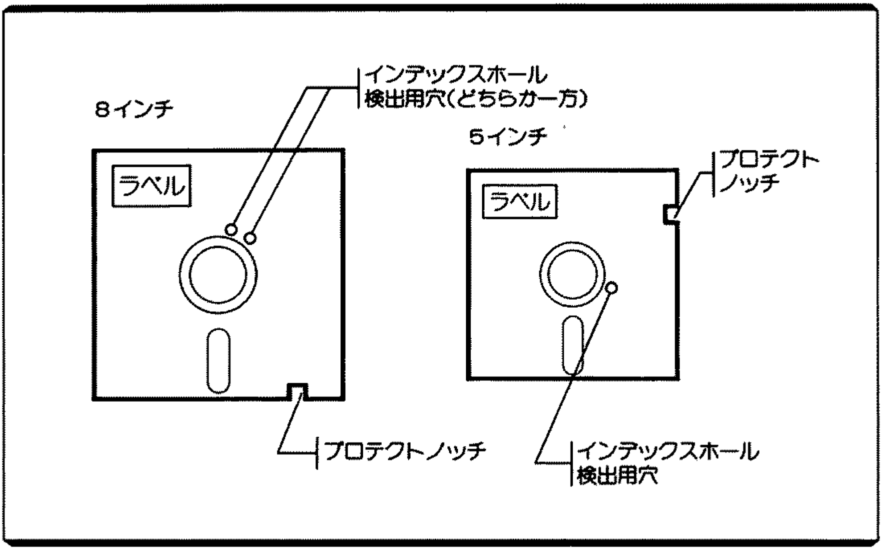
では、本当に完全なコピーは不可能なのでしょうか。

◎―――――――――― 完全なコピーは不可能か

完全に同じものを作るとはどういうことでしょうか。その対象、ディスケットに注目します。ディスケットを構成する要素は２つであり、１つはジャケット、もう１つは中に入っている磁気円盤です。完全なコピーとはこのジャケットおよ

び磁気円盤について、外見も内面もまったく同じものを作ることを意味します。
１つひとつ検討を加えましょう。

図１．４　ディスケット



まず、ジャケットからです。ジャケットを決定する要素として考えられるものを列挙すると、それにはａ．大きさ、ｂ．形、ｃ．色、ｄ．ラベルなど数多くあります。１万円札のように複雑な色、模様を表面に印刷すれば、簡単には同じものが作れなくなります。「なるほどそうだけど、いったいどうやればそれをチェックできるのか」という声がでてきそうです。確かに、現在のドライブ装置ではジャケットの表面の模様を読み取ることはできません。人間が目で見ない限りチェックの方法はありません。この例は極端な例ですが、完全にコピーすることと、チェックできることとは別であることをわかっていただきたかったのです。

それでは、ジャケットに関して何をチェックできるでしょうか。

プロテクトノッチの有無、インデックスホール用の穴の位置（８インチのみ）ぐらいでしょうか。それぐらいなら充分に同じものを作れそうです。

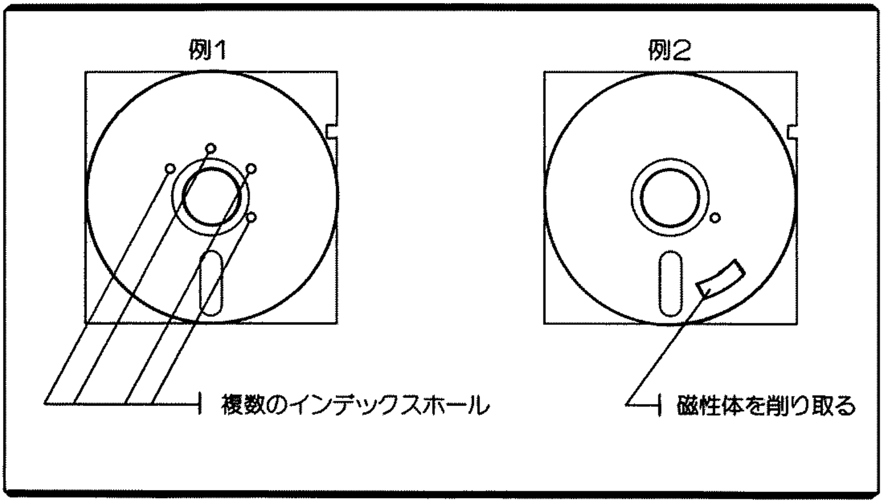
次は、磁気円盤です。これを決定する要素を大きく２つに分けて考えます。ハード的側面（外見）およびソフト的側面（内容）です。

ハード的側面に関しては規格で統一されていますから、通常は充分に同じものが得られます。しかし、規格を無視してよいのならまともに動作する範囲内でもかなりの変更が可能です。

(1) インデックスホールの数を複数にする。

(2) 表面の一部に磁性体を塗らない、または一部の磁性体を削り取る。

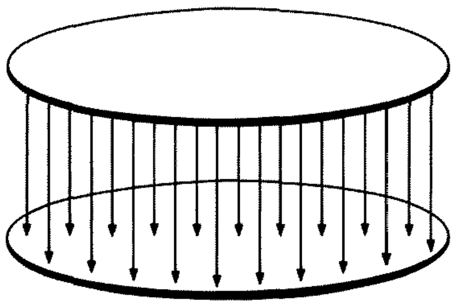
図１．５　ハード的な変更の例



(1),(2)の場合ともこれらを完全にコピーすることは位置の精度の点でかなり難しいでしょう。どこまで正確にその位置をチェックできるかに依存しますが、多分これらの方法を使った場合、コピーはほとんど不可能となるでしょう。

ソフト的側面、つまり磁気円盤上の磁気の記録状態に関してはどうでしょうか。円盤の両面には磁性体が塗ってあって、そこに磁気が記録されています。円盤上の磁気の状態をそっくりそのまま、もう１枚の円盤に写し取る装置があれば、コピーは完璧です。プロテクト側はお手上げとなります。

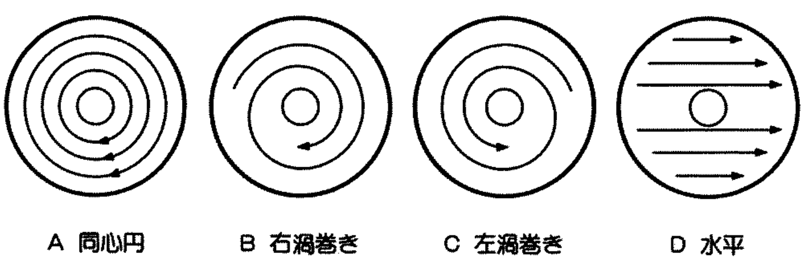
図１．６　磁気転写装置



幸い現在のところ、このような平面を平面のままコピーするような装置はありませんので助かっています。

平面のままではコピーができないので、普通はこれを線の集まりととらえます。もちろん、本物を作成する時も線の集まりとして書き込みますが、その際、決定すべき要素は多くのものが考えられます。

図１．７　線の流れ



(1)線の流れ

a．同心円.

b．右渦巻き

c．左渦巻き

d．水平

さすがに、dは実現不可能ですが、b，cは可能かも知れません。PC－9800では、通常aの同心円を使っています。この1つの円をトラックと呼びます。

(2)トラックの幅

(3)トラックの間隔

(4)トラックの絶対的位置

平面から線に分解するだけで、これだけの要素が浮かび上がってきます。コピーをするにはこれらの要素を合致させなくてはなりません。ところが（1)～(4)とも規格で統一されており、プロテクト側も今のところは規格どおりにしているようなので、現在は問題になっておりません。

ですから、これらは規格に従っているものと仮定しましょう。そうすると、各トラックを忠実にコピーできれば完全なコピーができることになります。

◎―――――――――― 各トラックの完全なコピーは可能か

これが、かなり難しいのです。唯一、かなり忠実にコピーできるのは、次のよな場合です。マスター（本物）とスレーブ（コピー）を、同一の軸上でインデックスホールの位置を合わせ回転させ、マスターから読み取った信号をそのままスレーブに書き込むのです。

もちろん、これには専用の装置がなくてはできません。さて、現実の環境ではどうかというと、2台のドライブ装置は別々の軸で回転しているので、同期をとることができませんから、1台のドライブ装置からいったん信号をメモリに読み込んだ後、もう1台のドライブ装置に書き込みを行なうことになります。

図1．8　同期コピー装置

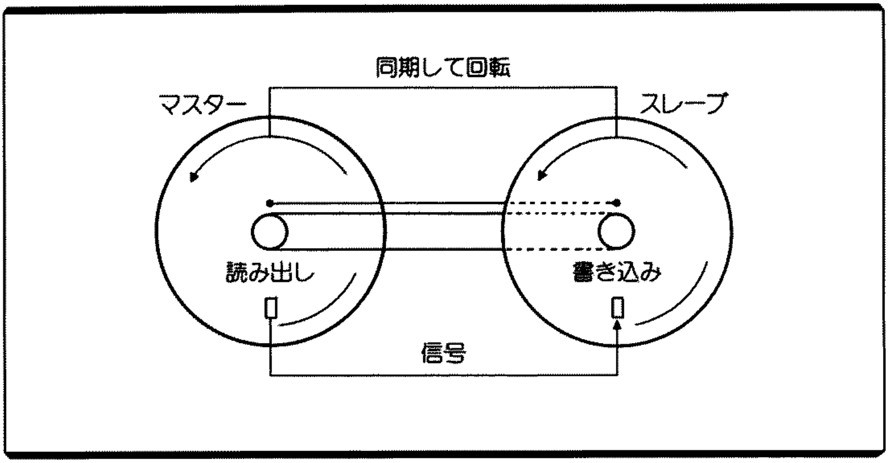
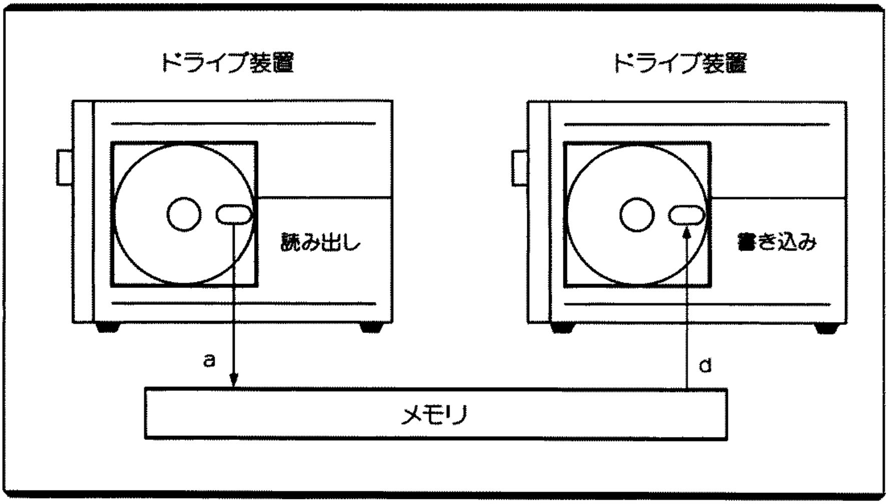


図1．9　メモリを介したコピー



ここで２つの大きな問題が生じます。

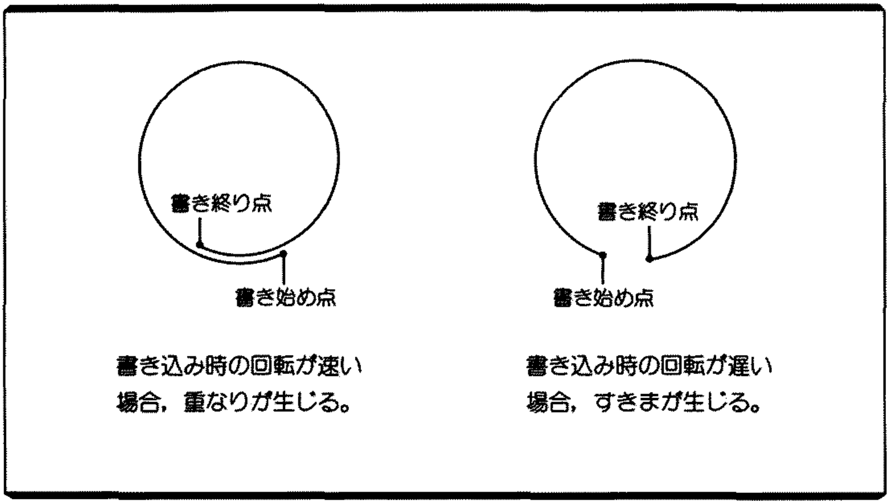
① 読み込み時と書き込み時の回転速度を完全に一致させることが難しい

② マスターから読み出した信号を忠実に書き込み時に再現することが難しい

まず、①の問題点について説明します。２台のドライブ装置は別々に回転しますから、当然、回転速度は異なります。ドライブ装置の規格では±２％程度の回転誤差は許容範囲とされていますから、ドライブ装置が異なればその程度の違いのあることは予想がつきます。また、たとえ同じドライブ装置であってもディスケットの摩擦状態によって変動し、また時間的にも変動します。ですから、同期して回転させない限り偶然に読み込みと書き込みがまったく同じ回転速度で行なわれることは、ないと言っても過言ではありません。

ところで、もしほんの少しでも読み込み時より、書き込み時の回転速度が速ければ、１周ちょうど書いたつもりでも、書き始めと書き終わりの部分が重なってしまいます。逆に、もし少しでも遅ければ、１周分書いてもすきまができてしまいます。

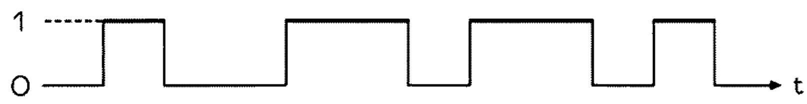
図１．１０　回転速度が異なると



いずれにしてもコピーは本物と違うものになってしまいます。

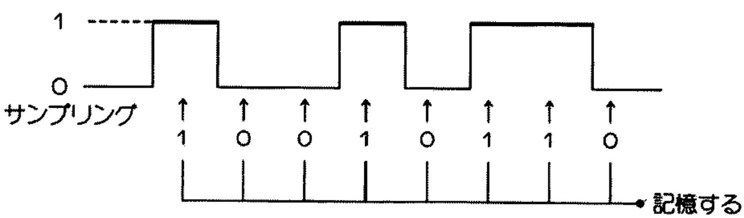
次に②の問題点について説明します。線状に書かれている磁気は、ヘッドを通して電気信号に変換されます。この変換された電気信号をメモリにたくわえておいて書き込み時に忠実に再生できればいいわけです。便宜的にこの電気信号は０と１の２レベルを持った信号と仮定します。

図１．１１　電気信号



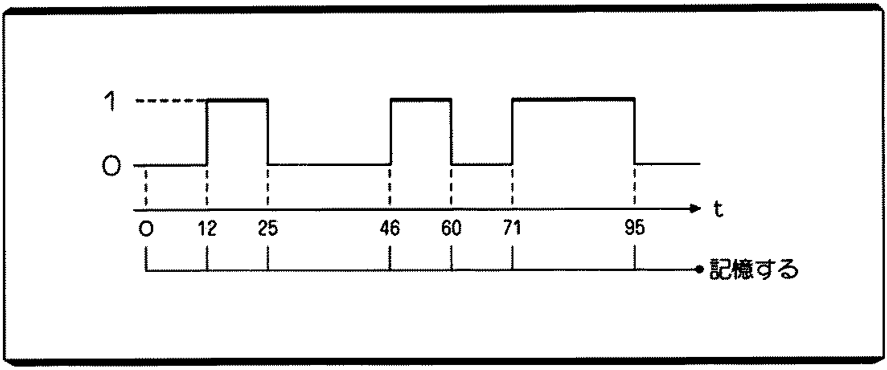
このような電気信号をメモリにたくわえる方法は２つあります。１つは一定時間ごとに信号が１か０かを読み取って（サンプリングという）記憶する方法です。

図１．１２　サンプリング方式



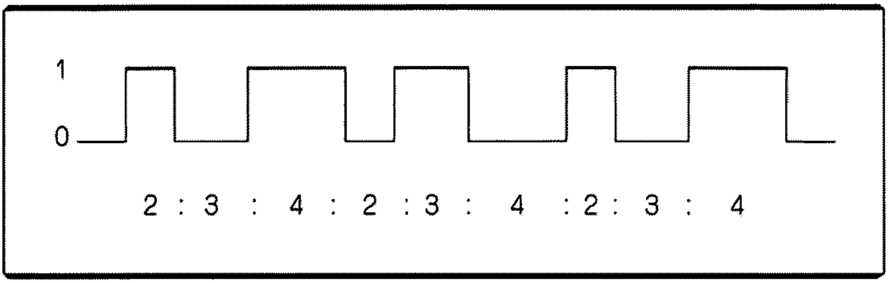
もう１つは信号が０から１、または１から０へ変化した時刻を記憶する方法です。　さて、前者はサンプリング間隔より細かい部分は記憶できないのに対して、後者は時刻測定の精度まで細かく記憶できます。現実に行なわれている方法は前者ですから、ここに１つ問題が生ずる可能性があります。

図1．13　変化点を記憶する方式



一方、フロッピディスクに書き込まれる信号はかなり規則性があるのも事実で簡単には次のようになっています。

図1．14　信号の規則性



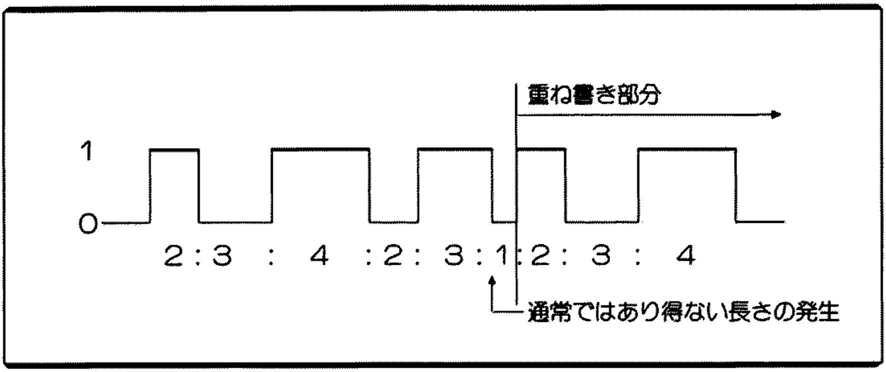
・0から1または1から0に変化するタイミングは必ず基本周期の2、3、4倍のいずれかである。

しかし、次のような場合があるので要注意です。

① トラックの一部分を重ね書きにした場合

確かに書き込み時には必ず先の規則に従うのですが、一部分を重ね書きにした場合その継ぎ目に関しては保証されません。

図１．15　不連続点



②　基本周期が２種類以上ある場合

これは実例をあげましょう。ＰＣ－９８０１ＦとＭの５インチディスクです。同じ５インチでＭはＦの1.5倍の記憶容量があります。両方ともトラック間隔は同じなので違いは基本周期にあるのです。つまり、Ｍの基本周期がＦの1.5分の１で、同じ長さのところに1.5倍の記憶ができるのです。

たとえ、このように基本周期が異なっていてもＦ、Ｍを使っている分には問題はありません。両方を混在させても一方しか読めないからです。ＶＭ/ＸＡなら使い道はありそうです。

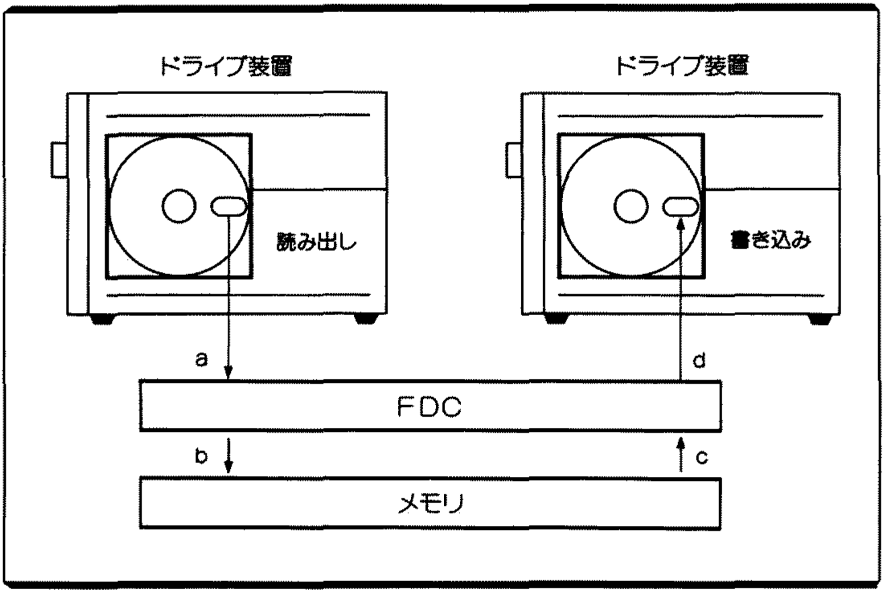
さて、いかがですか。今までみてきたのは，ヘッドから読み出された信号をサンプリングした後、直接メモリに取り込むことができた場合の問題点です。ＰＣ－９８００では，これは外付けハードウエアを付加した場合に初めて可能となります。忠実にコピーすることがいかに難しいかおわかり頂けたかと思います。

ＰＣ－９８００本体だけでは、忠実なコピーはさらに難しくなります。それは、ドライブ装置とメモリの間にＦＤＣ（フロッピディスクコントローラ）というくせものが介在してくるからです。

◎ くせもののFDC

ドライブ装置とメモリの間には、FDC（フロッピディスクコントローラ）と呼ばれるLSIが介在しています。このFDCがくせものです。なぜなら、FDCは読み込んだ信号の一部しかメモリには転送しないからです。

図1．16　FDCの介在



図のaのところでは、フロッピディスクから読み出された信号はすべてFDCに入ります。ところがFDCから出てくるbはaの一部のみです。つまり、読めない部分があることになります。

書き込みのときは逆にFDCに送ったデータ（c）よりも多くのデータ（d）がドライブ装置に送られます。つまりトラック1周全体のうち書き込むデータを自由に指定できない部分があるということです。

このように、FDCによる制約が大きいため読んだままを書き込むことはできないのです。これがプロテクトを成り立たせる1つの大きな要因です。

『究極のプロテクトは可能か？』　結論はあなた自身でお出しください。



2.1 FDCの知識
FDCの想定するフォーマット

ＰＣ－９８００シリーズでは、ＦＤＣ（フロッピディスクコントローラ）にμＰＤ７６５Ａという専用ＬＳＩが使われています。ディスクへのアクセスはすべてこのＦＤＣがつかさどっています。プロテクトをかけるときには専用のハードウエアを使うこともできますが、本物かどうかのチェックをするときにはＰＣ－９８００シリーズそれ自身で行なわなくてはなりませんから、このＦＤＣの存在は無視できません。強いプロテクトができるかどうかは、ＦＤＣの能力をいかに引き出すかにかかっていると言ってもよいでしょう。ＦＤＣの知識はプロテクトの基本です。本章はじっくりお読みください。

ＦＤＣはディスクのアクセスに必要な読み書き、フォーマットなどの操作を簡単にすることを目的として作られたＬＳＩです。μＰＤ７６５ＡはＩＢＭトラックフォーマットというフォーマットを想定して作られています。ですから、これに準ずるフォーマットであれば、いとも簡単にフォーマットできます。数十バイトのデータと数個のパラメータを用意するだけで、１トラック数千バイトの書き込みをしてくれます。ところが、このフォーマットに従わないとなると、突然、困難になります。普通は、親切なこの機能もプロテクト屋にとっては、おっせかいとしか言えません。

図２．１がＦＤＣの想定するフォーマットです。インデックスホールの位置で円状のトラックを切り、まっすぐに伸ばして書いたものです。実際は図の１番下と上がつながってループになっています。

図２．１　ＦＤＣの想定するフォーマット

区分		フィールド	長さ（バイト） 倍密	長さ（バイト） 単密
プリアンブル		GAP0	80	40
プリアンブル		SYNC	12	6
プリアンブル		IM	4	1
プリアンブル		GAP1	50	26
1セクタ	ID部	SYNC	12	6
1セクタ	ID部	IAM	4	1
1セクタ	ID部	ID	4	4
1セクタ	ID部	CRC	2	2
1セクタ		GAP2	22	11
1セクタ	データ部	SYNC	12	6
1セクタ	データ部	DAM or DDAM	4	1
1セクタ	データ部	DATA	128×2^n	128×2^n
1セクタ	データ部	CRC	2	2
1セクタ		GAP3	1~256	1~256
1セクタ				
		⋮ セクタの繰り返し ⋮		
ポストアンプル		GAP4	トラック全体ー上記合計	トラック全体ー上記合計

（全体で1トラック）

トラックの先頭にはプリアンブルと呼ばれるものがあり、次にセクタが続きます。１つのセクタはＩＤ部とデータ部からなり、以後必要なだけセクタが繰り返され、最後にポストアンブルがつきます。次に各部の説明に移ります。

◎ 各部の説明

プリアンブル

フォーマットのときには必ず作成されるが、読み書きにはまったく必要のない部分。この部分は破壊してしまってもまったく影響を与えない。ただ、トラックの先頭から少しの区間ではＩＤを読めないので、そこにＩＤ部がこないように避けるために役立っている。

セクタ

セクタはＩＤ部とデータ部からなる。普通、プログラムやデータなどの保存に使えるのは、このデータ部の中のＤＡＴＡという部分だけである。全体の３分の２位である。アンフォーマット容量（１トラック全体）に対して、フォーマット容量（ＤＡＴＡ部分のみ）がかなり小さくなるのは当然である。

ＩＤとはセクタの名前のようなもので、データ部の読み書きはこれをキーに指定する。

ポストアンブル

セクタをいくつか書いたときトラックの１周まで、まだ余裕があればその部分は埋めつくされるが、それがポストアンブルである。プリアンブルと同様、読み書きにはまったく必要とされない。

ＧＡＰ

ＧＡＰ（ギャップと読む）は、場所によって０から４と名前がつけられている。ＧＡＰ０とＧＡＰ１はプリアンブルに、ＧＡＰ２は各セクタのＩＤ部とデータ部

の間に、ＧＡＰ３はセクタ間にある。ＧＡＰ４はポストアンブルそのものである。本来の目的はデータなど何度も重ね書きをした場合、多少位置や長さがずれても他に影響を与えないようにするための緩衝帯である。書かれる値はいずれも倍密度では４ＥＨの連続、単密度ではＦＦＨの連続である。書かれる長さは図２．１に示すとおりで、ＧＡＰ３のみフォーマット時のパラメータで長さを指定できる。ＧＡＰ０～ＧＡＰ２は常に固定、ＧＡＰ４は自動的に余っている部分の長さとなる。

表２．１　ＧＡＰに書かれる値

倍密度	単密度
4E	FF

ＧＡＰ２を除いてＧＡＰに書かれている値、長さは読み書きに影響を与えない。

表２．２　ＧＡＰの長さの許容範囲

	正常にリードできる限界
GAP0	制限なし
GAP1	制限なし
GAP2	標準値の−20%から+150%まで この範囲を越すとそのセクタのデータを読めない。
GAP3	制限なし、ただし、極端に短い場合、重ね書きをしたデータが後方へずれると次のセクタのＩＤ部を破壊する可能性はある。
GAP4	制限なし

ＳＹＮＣ

ＶＦＯの周波数を回転に同期させるためにある。プリアンブルと各セクタのＩＤ部、データ部の先頭にそれぞれあるが、プリアンブル以外のＳＹＮＣは重要である。ＳＹＮＣがなければ、それぞれＩＤ部、データ部が読めない。ＳＹＮＣに書き込まれている値は、倍密度、単密度とも００である。長さも重要で標準値より30％以上短くなると読めなくなる。

IM（インデックス・マーク）

トラックの先頭をソフト的に示すためのマーク。μPD765Aは、IBMトラックフォーマットと互換性を持たせるため、フォーマット時には作成するが実際にはまったく使用していない。トラックの先頭は常にインデックスホール信号で検出する。IMに書かれている値は次のとおり。

表2．3　IMの値

	値	備　考
単密度	FC	ただし、ミッシングクロック データ=1111011101110101
倍密度	C2 C2 C2 FC	ただし、C2はすべてミッシングクロック データ=0101001000100100

IAM（IDアドレス・マーク）

ID部であることを示すマーク。FDCはSYNCに続くIAMを検出すると次にはIDが続くとみなす。このIAMもミッシングクロックが含まれており、通常のデータと区別される。IAMに書かれている値は次のとおり。

表2．4　IAMの値

	値	備　考
単密度	FE	ただし、ミッシングクロック データ=1111010101111101
倍密度	A1 A1 A1 FE	ただし、A1はすべてミッシングクロック データ=0100010010001001

ID

4バイトのデータでセクタを特定するための名前のようなもの。それぞれ1バイト毎にC、H、R、Nと呼ばれる。それぞれ、

C＝シリンダ番号
H＝ヘッド番号
R＝セクタ番号
N＝セクタ長コード

を意味することになっているが、Ｎを除いては、必ずしも実際の物理的な番号と一致している必要はなく、読み書きのときに同じ値を指定しさえすればよい。このことは、C、H、Rは単なるデータであることを意味する。Nに関しては、読み書きのときに指定するパラメータが実際の読み書きのセクタ長になるが、それは、読み書きしたいセクタのＩDに書かれているNと一致せねばならず、結局、Nに限り、読み書きするときのセクタ長コードをＩDに書いておかなくてはならない。Nとセクタ長の関係を次に示す。

表2．5　Ｎとセクタ長の関係

N	0	1	2	3	4	5	6	7	8以上
セクタ長	128	256	512	1024	2048	4096	8192	16384	32768

ＣＲＣ

Cyclic Redundancy Codeといって読み込み時のエラー検出に使われる。ＩD，DATAのすぐ後にあり、それらのデータすべてからある計算式で求めた2バイトの値である。機械語入力のミスの発見に使われるチェックサムみたいなものである。FDCは読み込み時には必ずこのCRC検査を行なって読み取りエラーがないかどうかをチェックしている。ＩDのCRC検査でエラーとなるとそのセクタに対する読み書きはできない。DATA後のCRC検査ではエラーになっても一応データは読めている。ただし、信頼性はない。

DAM（データアドレスマーク）および DDAM（ディリーティッド・データアドレスマーク）

IDの始めをIAMが示すように、DATAの始めを示すのがDAMまたはDDAMである。DDAMの先頭のDは DeletedのDであるが、決してデータが消されてしまってなくなったわけではない。DDAMのついているセクタにもちゃんとデータを入れることができる。DAMとはマークの値が違うだけである。つまり、各セクタに1ビット余分に情報を付加することができるに過ぎない。最初、フォーマット時にはDAMが付いている。そのセクタにデータを書き込むときに、Write Deleted Data というコマンドを使うことによって初めてDDAMは作られる。

DAM，DDAMの値は次のとおり。これらもミッシングクロックを含むので普通のデータとは区別される。

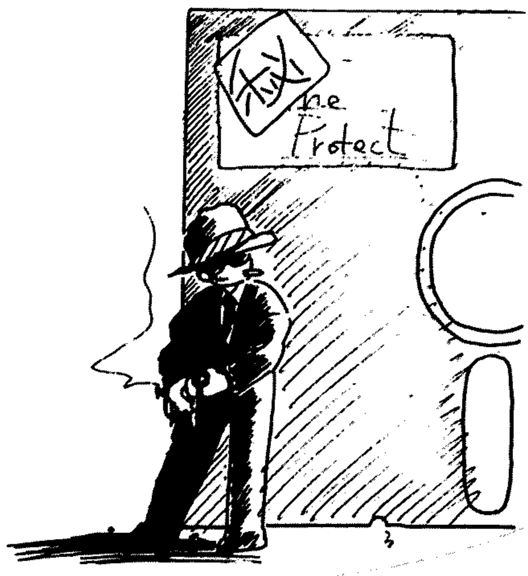
表2．6　DAM，DDAMの値

		値	備　考
単密度	DAM	FB	ただし、ミッシングクロック データ＝1111010101101111
	DDAM	F8	ただし、ミッシングクロック データ＝1111010101101010 1
倍密度	DAM	A1 A1 A1 FB	ただし、A1はすべてミッシングクロック データは表2．4と同じ
	DDAM	A1 A1 A1 F8	ただし、A1はすべてミッシングクロック データは表2．4と同じ

DATA

Writeコマンドで唯一任意の値を書き込める領域。DATAの長さはフォーマット時には Write IDコマンドで与えるパラメータのNに基づき表2．5のように決まる。ところがDATAを読み込む Readコマンドや、実際にデータ

を書き込むWriteコマンドでは、IDに書かれているNに従ってDATAの長さを決めるため、両者の間に違いがある場合は、フォーマット時と異なるDATA長で読み書きが行なわれることになる。



2.2 FDCの知識 FDCのコマンドを検討する

μＰＤ７６５Ａは実に16種類ものコマンドを持っています。

表２．７　コマンドの一覧

コマンド	機　能
Read Data	データを読む
Read Deleted Data	ディリティッドデータを読む
Write Data	データを書く
Write Deleted Data	ディリティッドデータを書く
Read ID	ＩＤを読む
Write ID	フォーマットする
Read Diagnostic	トラックの先頭からＩＤに関係なくデータを読む
Scan Equal	メモリとディスクのデータの一致を調べる
Scan Low or Equal	同様でディスクのデータが小さいときも可
Scan High or Equal	同様でディスクのデータが大きいときも可
Seek	ヘッドを目的のシリンダまで移動させる
Recalibrate	ヘッドを0シリンダまで移動させる
Sense Interrupt Status	シークの結果を読み出す
Sense Device Status	ディスクの状態を読み出す
Specify	ＦＤＣの時間関係を設定する
Invalid	未定義命令

各コマンドのパラメータの一覧は付録3にあります。

◎ マルチセクタとＴＣとマルチトラック

Read Data，Read Diagnostic，Write Data，Write Deleted Dataなどのコマンドには、いったんデータ転送を始めると１セクタ分の処理が終わっても、やめろというまで次のセクタ、その次のセクタへと処理を続ける性質があり、これをマルチセクタ機能と呼びます。セクタを終わる度にセクタ番号は自動的に１ずつ増やされていきます。ではどうしたら終わってくれるかというと、次の３つの場合があります。

① セクタ番号がＥＯＴに到達

セクタを移る度にセクタ番号（Ｒ）は１ずつふやされていきますが、それがＥＯＴと一致したら、そのセクタを処理後、終了します。つまり極端な場合、読み書きを始めるセクタ番号よりも小さい値をＥＯＴにセットしておけば、この条件では終わらないことになります。

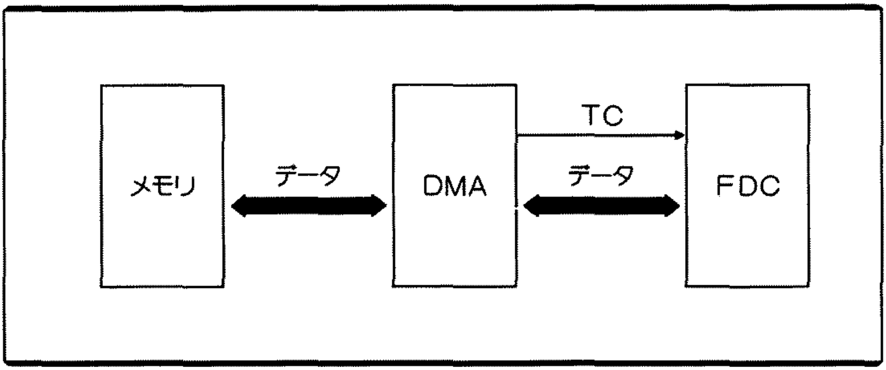
Read Diagnosticの場合は、ＩＤの値が無視されるのでセクタ番号に関係なく、読み込んだセクタの個数がＥＯＴに達したときに終了します。

なお、マルチトラックフラグ（ＭＴ）が１の場合は、ヘッド０を処理中に限り、この条件は終了せず、ヘッドが１に代わり、セクタ番号が１にリセットされ、処理を続行します。ヘッド１から０への連続は行なわれません。

② ＴＣの入力

ＦＤＣにはＴＣ（Terminal Count）と呼ばれるピンがあり、ここに信号が入力されると、現在処理中のセクタで終了します。ＰＣ－９８００シリーズではこのＴＣはＤＭＡＣに接続されており、ＤＭＡＣにあらかじめセットされている転送バイト数の転送が終わるとＤＭＡＣはＴＣを出力し、終了を促します。ＴＣの入力による終了が普通の終了方法です。

図2．2　TCの入力



③　エラーの発生

リード、ライト処理中にエラーが発生したときも終了の原因になります。Read Dataで、ディリーティッドデータを読もうとしたとき、および、その逆にRead Deleted Dataで普通のデータを読もうとしたときにも、エラーとなり終了しますが、SK（スキップ）が1の場合はそのセクタが読み飛ばされエラーにはなりません。

◎――――――――――――――――――――各パラメータの意味

MT（マルチトラック）

マルチセクタ機能をヘッド0からヘッド1に連続して行なうことを指定する。
1：ON，0：OFF

MF（MFM／FM）

倍密度か単密度かを指定する。

1：倍密度、0：単密度

ＳＫ（スキップ）

Read Dataコマンドで、デイリーテイッドデータをリードするとき、飛ばしてリードすることを指定する。同様にRead Deleted Dataコマンドでは、普通のデータを読み飛ばす。

１：ON，０：OFF

ＨＤ

物理的ヘッド番号、0 or 1。ディスクの表面か裏面かは、唯一このビットで指定する。パラメータＨは物理的ヘッドには関係ない。

ＵＳ

μＰＤ７６５Ａでは、１つのＦＤＣに対して４台までドライブ装置をつなげられる。このＵＳでそのうち１つを選択する。

Ｃ

シリンダの意味ではあるが物理的なシリンダとはまったく関係なく、値は0～255のいずれでもよい。セクタを指定するときの４バイトの内の１バイトとして使われる。Read、Writeは、Ｃ、Ｈ、Ｒ、Ｎの４バイトが一致しないと行なわれない。物理的なシリンダ番号はSeekコマンドのＮＣＮによって指定する。

Ｈ

ヘッドの意味ではあるが物理的なヘッドとはまったく関係なく、値は0～255のいずれでもよい。セクタを指定するときの４バイトの内の１バイトとして使われる。物理的なヘッド番号はパラメータＨＤによって指定する。

Ｒ

セクタ番号を意味する。値は0～255のいずれでもよい。マルチセクタ機能ではこの値を基にインクリメントされる。

Ｎ

セクタ長を意味する。この値によって処理されるセクタ長は決まる。Ｎの値とセクタ長（ＤＡＴＡ部分）との関係は表２．５に従う。また、Ｃ、Ｈ、Ｒと同様にセクタを指定する１バイトとしても使われる。Write ID ではこのＮでセクタ長が決まる。

ＥＯＴ

処理する最終のセクタ番号を指定する。ＦＤＣはマルチセクタ機能によって、ＴＣを入力しない限りセクタ番号を１ずつふやしながら処理を続けるが、ＥＯＴと一致したところで、そのセクタの処理後コマンドは終了する。

ＳＣ

Write ID コマンドで作成するセクタの数を指定する。ただし、途中でＴＣが入力された場合は、それ以降新しいセクタは書き込まれない。１トラックに入り切らない数のセクタを指定しても、ＦＤＣはおかまいなく最後まで書き込みを実行する。

ＧＰＬ

Write ID とその他のコマンドで意味が異なる。Write ID では書き込まれるＧＡＰ３の長さを意味するが、その他のコマンドではＤＡＴＡ後の読み飛ばす長さを意味する。つまり、連続してリード、ライト等を行なうとき、ＤＡＴＡに読み書きをした後、ＧＰＬの長さ分の直後のデータを読み飛ばしてからＩＤのサーチをする。そのため、ＧＰＬをＧＡＰ３より長くすると続くセクタのＩＤ部が読み取れなくなるので、普通はＧＡＰ３の半分程度に設定する。もっと短くても構わないとも思われるが、ＤＡＴＡを書きかさねるとＤＡＴＡ直後の部分ではビットずれが生ずるので、この不安定な部分がＦＤＣに誤動作をさせないようにこの部分を読み飛ばすのである。０を指定すると256を意味する。

ＤＴＬ

１セクタ中で処理するバイト数を指定する。単密度かつＮ＝０でリード、ライトするときのみ意味を持つ。

Ｄ

Write IDのときにＤＡＴＡを埋める１バイトデータを指定する。

次にＦＤＣの主要なコマンドについてさらに検討を加えます。

◎── Read Diagnostic（リード・ダイアグノスティック）

このコマンドはトラック中のほぼ全データを読むことができる、非常に便利なコマンドです。普通の Read はＩＤ部のＣ、Ｈ、Ｒ、Ｎのすべてが一致しない限りそのデータ部を読むことができませんが、Read Diagnostic では、ＩＤ部のＣ、Ｈ、Ｒ、Ｎに関係なくそのデータ部を読むことができます。読み込まれるセクタ長は実際のセクタ長に関係なく、パラメータで指定するＮで決まりますから、Ｎを大きくしてやれば、実際のデータ部をこえて読み込むことも可能で、ＧＡＰ部も読めるわけです。

パラメータで指定するＮとセクタ長との関係は原則として表２．５に従いますが、１つ例外があって倍密度でＮ＝０のときは80バイトとなります。

１トラック全体の長さが５インチ２ＤＤで６千数百バイト、２ＨＤまたは８インチで１万数百バイトであることから、このコマンドを用いれば充分に１トラック全体を読み込めることがわかります。また、このコマンドのもう１つの特徴は必ず、トラックの先頭のセクタから読み込みが開始されることです。この特徴を使えばトラックの先頭のセクタを見つけるのにも使えます。

一つ注意すべきことは、先頭のセクタにＤＡＭ（データアドレスマーク）またはＤＤＡＭがない場合は、ただちにエラーとなるため、このコマンドが使えないことです。これが、このコマンドの弱点です。ＤＡＭをなくす方法は、Write

のところで述べます。

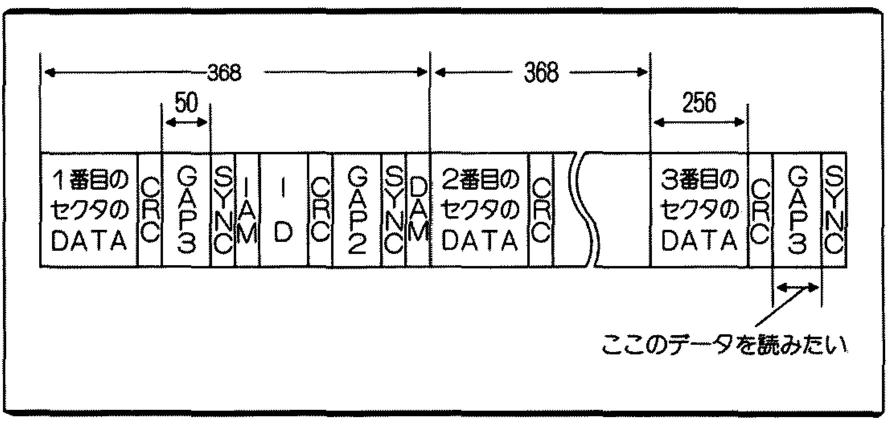
コマンドの動作

1．トラックの先頭からＩＤをサーチする。
2．初めて見つかったＩＤの次のデータ部から指定したセクタ長分のデータを読み出す。
3．ＤＭＡＣからＴＣが来ない場合は、ＧＰＬ分の長さを読み飛ばした後、ＩＤをサーチし、２から繰り返す。
4．読み出したセクタの数がＥＯＴに達した時はそこで終了する。ただし、現在のヘッドが０で、マルチトラック指定をしている場合はヘッドが１に変わり、続けて読み出される。
5．一つもＩＤが見つからないまま２回インデックスホールを検出した場合は、その時点でエラー終了する。

例題

① １セクタ256バイト、ＧＡＰ３が50バイトのとき、３番目のセクタと４番目のセクタのＧＡＰ部を読みたい。どうすればよいか。

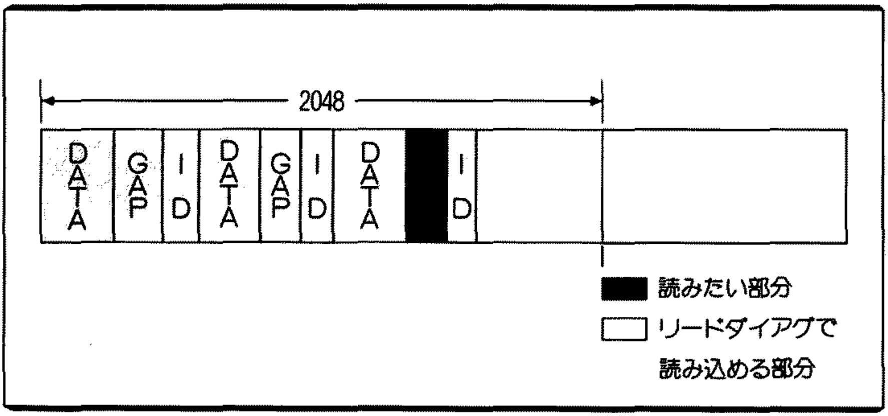
図２．３　例題１



解　方法は２とおりあります。

Ａ．１番目のセクタからＮ≧４でリードダイアグする。

図２．４　Ｎ＝４でリードダイアグ



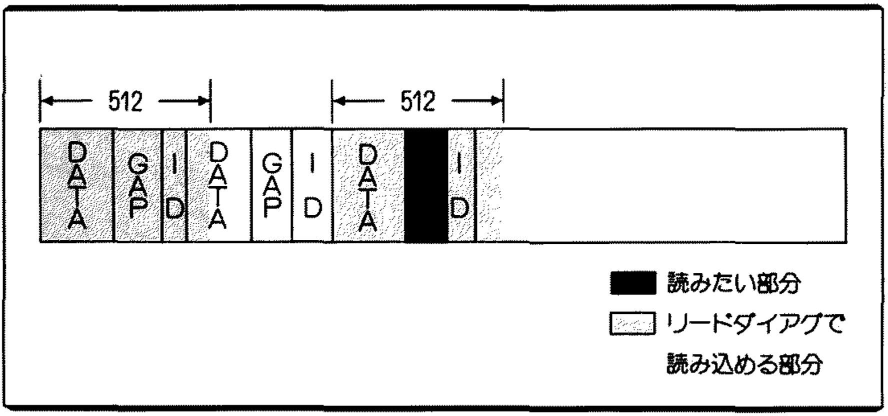
目的の場所は先頭から数えて992～1043バイト目であるから、Ｎ≧４なら先頭から読んでもこの部分を読むことができる。この方法の欠点は途中に３つもセクタを挟むことで、まったくＤＡＴＡを重ね書きしないならばよいが、ＤＡＴＡを重ね書きした場合は合計５箇所もビットずれを起こす可能性があり、目的の部分の位置が正確でなくなる。

Ｂ．Ｎ＝２、ＧＰＬ≦100でリードダイアグする。マルチセクタ機能を使って２セクタ分読む。

Ｎ＝２の場合は、ＦＤＣは512バイトＤＡＴＡがあると思って読み始める。１セクタ分（512バイト）読み終えたところでヘッドは実際の２番目のセクタのＩＤを通り越しているので、次に読まれるのは、３番目のセクタのＤＡＴＡからである。またＧＰＬを長くしすぎると、３番目のセクタのＩＤまで読み飛ばしてしまうので、166バイト以下にしなくてはならない。この方法ではＡと違ってビットずれが生ずる可能性のあるのは３番目のＤＡＴＡ直後の１箇所のみであり、

かなり信頼度は増す。さらに3番目のセクタのみ重ね書きをしなければ、100％完全なデータを読むことも可能となる。

図2．5　N＝2でリードダイアグ

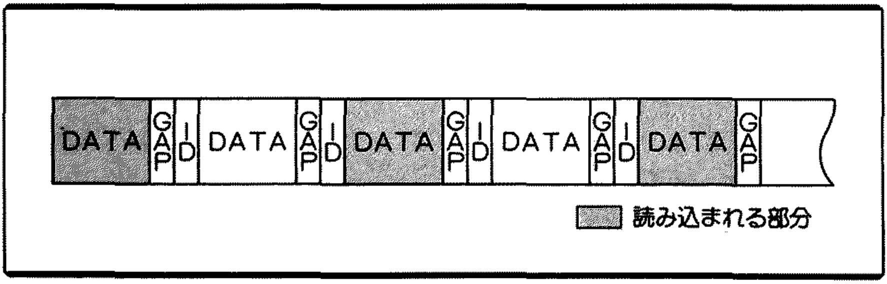


② 1セクタ256バイト、GAP3が50バイトのとき、1番目、3番目、5番目のセクタのデータ部だけを順に読みたい。どうすればよいか。

解

N＝1、GPL>50でリードダイアグする。マルチセクタ機能で3セクタ分読む。

図2．6　大きいGPLでリードダイアグ



リードダイアグは、ＩＤが一致するしないにかかわらず読めるのが特徴だが、必ずしも先頭から連続にではなく１つおきに読むことができることを示す。１番目のセクタを読んだ後、２番目のセクタを読まないようにするには、ＧＰＬをＧＡＰ３より長くして次のＩＤを読み飛ばせばよい。例題ではＧＡＰ３は50バイトであるから、余裕をみてそれより充分に長くすればよい。例えば200バイト（最大は256バイト）。

◎―――――― Write ID（フィジカル・フォーマット）

このコマンドは唯一ＩＤを書き込むことができるコマンドです。データの読み書きはすべてディスケット上に書かれているＩＤを元にして行なわれますから、最初に必ずＩＤを書き込んでおかなくてはなりません。この作業をフォーマットと呼びます。このコマンドはよくできているので、少数のパラメータを与えるだけで、１トラック全体のフォーマットをしてくれます。逆に言えば、一定のパターン（図２．１）でしかフォーマットできないことを意味します。

変更可能なのは次の５つです。

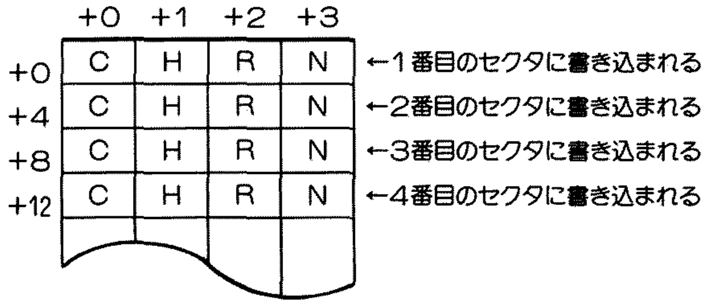
① セクタ数　　　　　　　　：ＳＣ
② ＤＡＴＡの長さ　　　　　：Ｎ
③ ＧＡＰ３の長さ　　　　　：ＧＰＬ
④ ＤＡＴＡを埋めるバイトデータ：Ｄ
⑤ ＩＤに書き込む値　　　　：転送するデータによる

当然、次のようなことはできません。

① セクタによってＤＡＴＡの長さを変える
② セクタによってＧＡＰ３の長さを変える
③ ＧＡＰ３の長さを256バイトより大きくする
④ ＤＡＴＡの中をいろいろなデータで埋める

IDに書き込むデータは次のように用意します。

図2．7　IDに書き込むデータの構成



コマンドの動作

1．トラックの先頭から上記のパターンに従ってデータを書き込む。

2．書き込んだセクタの数がSCに達するか、TCを受けるまでセクタパタンを繰り返す。

3．最後のセクタの書き込み中（厳密には、ID前のSYNCからDATA後のCRCまで）にインデックスホールを検出した場合は、GAPデータを1バイトだけ書き込んで終了する。検出しなかった場合はインデックスホールを検出するまで、GAPデータを書き込む。

◎ Read ID（リードID）

ID部を正確に読むことができる唯一のコマンドです。Read Data では、IDが完全に一致しない限りデータを読むことができません。そこで、データを読むにも、まず、Read ID を使ってIDを知る必要があります。Read ID は、Read Diagnostic と違ってトラックの先頭からではなく、コマンドが与え

られた時点のヘッドのある場所からＩＤをサーチします。ですから、これだけではどのセクタのＩＤを読んだのかわかりませんから、絶対的な位置が必要な場合は、後に述べる方法によって最初にトラックの先頭に、ヘッドを位置づけてやります。

コマンドの動作

1．現在ヘッドのある位置からＩＤをサーチする。
2．初めて見つかったＩＤのＩＤ部を読み取り、すぐ終了する。
3．一つもＩＤが見つからないまま、２回インデックスホールを検出した場合は、その時点でエラー終了する。

<ヘッドをトラックの先頭に位置づける方法>

Read ID，Read Diagnostic，Read Data などで、ＩＤが一つも見つからなかった場合は２回インデックスホールを検出した時点で終了するという性質を利用します。インデックスホールを検出した直後にはヘッドはトラックの先頭にあるからです。問題はＩＤが一つも見つかってはいけないことで、わざとＩＤが見つからないようにコマンドを与えます。記録密度の違いを利用するのです。つまり、記録密度の違うものを読もうとしてもまったく読めないという性質を利用します。もし、実際が倍密度で書かれているなら、単密度でリード系コマンドを行ない、単密度ならその逆をすればいいのです。ただし、単密度で書かれているものを倍密度で読もうとするとＦＤＣがハングアップすることがありますから、注意してください。

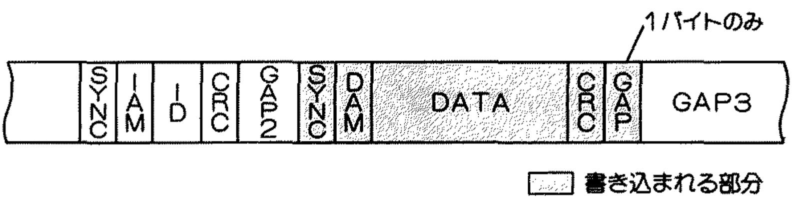
◎── Write Data（ライト・データ），Write Deleted Data

データ部に任意の値を書き込むことができるコマンドです。書き込みはパラメータで指定するＩＤと等しいＩＤ部を見つけた後、直ちに行なわれます。書き込

まれるデータ部の長さはパラメータのN（つまり、ＩD部のN）に従い、フォーマット時に作られたデータ部の長さには関係ありません。極端な場合、ＩD部だけ存在すれば、データ部のないセクタにも書き込みは行なわれます。フォーマット時より長いデータを書いた場合は、続く部分がつぶされてしまうことはいうまでもありません。

このコマンドで書き込みが行なわれる部分を図に示します。

図2．8　書き込みが行なわれる部分



書き込み開始はＩD部に続くGAP２の直後、ＳＹＮCの先頭からで、書き込み終了は、CRCの後に１バイトのGAPを書いたときです。このコマンドはあくまでもトラック中の一部分に重ね書きをするわけですから、完全にピッタリ重ならない限り、書き込み開始位置と終了位置にはビットずれができることを認識しておいてください。

Write Data と Write Deleted Data の違いは、データ部直前に書かれるアドレスマークが、DAM（FB）かDDAM（F８）かの違いだけです。この違いは、 Read Dataコマンド時にリザルトステータスとして検出できます。

データ部の長さはNに従いますが、倍密度でNが０（128バイト）のときには注意が必要です。書き込みは行なえますが、このときは、アドレスマークが書かれません。このセクタは２度と読めなくなるだけでなく、このセクタがトラックの先頭にある場合は Read Diagnostic も使えなくなります。Nとデータ長との関係は表2.5のとおりです。Write ID と同じくデータ長をトラック長よりも長くすれば、自分自身のＩDまでつぶしてしまいます。

コマンドの動作

1．現在ヘッドのある位置からＩＤをサーチする。
2．初めて見つかったＩＤのＩＤ部を読み取り、指定したＩＤと一致するかを調べる。一致するＩＤが見つかるまで繰り返す。
3．見つかった場合ＩＤのＣＲＣチェックをする。ＣＲＣエラーならその時点でエラー終了。
4．ＣＲＣがＯＫなら、一定バイト読み飛ばした後、ＳＹＮＣ，ＤＡＭ（またはＤＤＡＭ），データ，ＣＲＣ，ＧＡＰ１バイトを書いて終了。
5．一つも一致するＩＤが見つからないまま、２回インデックスホールを検出した場合は、その時点でエラー終了する。

◎ Read Data

データ部を読み込むコマンドです。Read Diagnosticとは異なり、読み込みはパラメータで指定するＩＤと一致するＩＤ部を持つセクタに対してのみ行なわれます。万一、１トラック上に同じＩＤが複数存在する場合は、少しやっかいです。厳密には、コマンドが与えられた時点にヘッドのある位置からＩＤをサーチし、最初にＩＤが一致したセクタが読み込まれます。ですから、それらのうちの特定のセクタを読みたい場合にはコマンドを与えるタイミングを調整してやらねばなりません。

読み込まれるデータ部の長さはパラメータのＮ（つまり、ＩＤ部のＮ）に従い、フォーマット時に作られたデータ部の長さには関係ありません。実際のセクタ長より長いＮをＩＤ部に書いておけば、そのセクタに続くＧＡＰ部まで読むことができます。極端な場合そのセクタを先頭に１トラック全体を読むことも可能です。

コマンドの動作

1．現在ヘッドのある位置からＩＤをサーチする。
2．初めて見つかったＩＤのＩＤ部を読み取り、指定したＩＤと一致するかを調べる。一致するＩＤが見つかるまで繰り返し。

3．見つかった場合ＩＤのＣＲＣチェックをする。ＣＲＣエラーならその時点でエラー終了。

4．ＣＲＣがＯＫなら一定バイト読み飛ばした後、ＤＡＭ（またはＤＤＡＭ）をサーチし、ＤＡＭまたはＤＤＡＭが見つからなければエラー終了。

5．見つかったら続くデータを読み取り転送する。Ｎで示されるバイト数を読み込んだ後、ＣＲＣチェックを行なう。

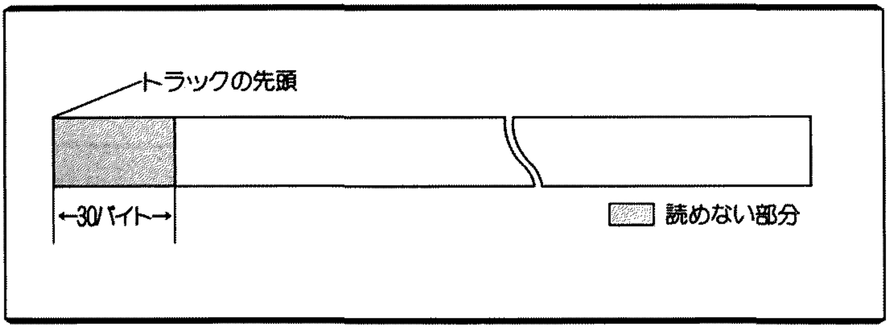
6．一つも一致するＩＤが見つからないまま２回インデックスホールを検出した場合は、その時点でエラー終了する。



2.3 FDCの知識
読めない部分があるってホント？

では、次にμＰＤ７６５Ａの持つ特殊な性質について整理しておきましょう。一つは１トラックの中で読めない部分があるということです。トラックの先頭から少しの区間（倍密度で約30バイト、単密度で約15バイト）にＩＤがあると、そのＩＤを読むことができないのです。厳密にはＩＡＭが認識できないようです。ＩＤとしては読めないのですが、データとしては読めますから、リードダイアグなどを使ってトラックの１周以上を読み込めば、その部分に書かれているデータは読むことができます。

図２．９　この部分にＩＤがあってもＩＤとして認識できない



通常この部分はプリアンブルが書かれているところで、普通は無視しているところですから、読めなくてもまったく問題ないわけです。

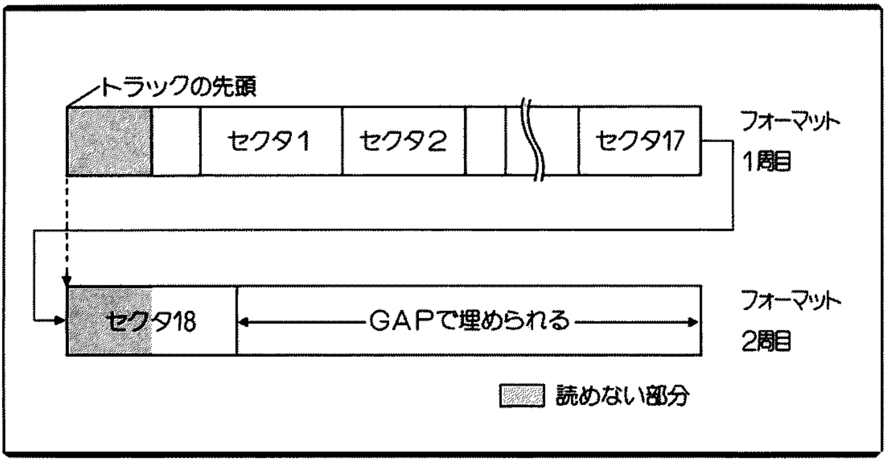
この性質自体はあまりプロテクトには利用されませんが、プロテクトでは異常なフォーマットを使うことも多いので、この部分にＩＤをもってきたら読めなくなることを理解しておく必要があるでしょう。

◎―――――― ＩＤがあるのにアンフォーマット

この性質は実際にはどういう現象として観測されるかを考えてみましょう。

まず、この読めない部分だけにＩＤを作ります。Write ID で1周以上のフォーマットをすれば可能ですね。例えば、２ＤＤで256バイトのセクタを18個、ＧＡＰ３を71バイトでWrite IDします。18番目のセクタから2周目に入り、ちょうど読めない部分にＩＤが作成されます。

図２．10　ＩＤがあるのにアンフォーマット



18番目のセクタはトラックの先頭より少し後から始まっていますから、Write ID の性質により18番目のセクタ書き込み後、インデックスホール検出まで残りの部分はＧＡＰで埋められ、結局、1〜17のセクタは塗りつぶされてなくなってしまいます。残ったのは18番目のセクタのみですが、このセクタはＩＤを読めないのでRead ID を使っても検出できず、アンフォーマットとみなされます。もちろん、このセクタに対してはリードもライトもできません。

◎プロテクトへの可能性

さて、この性質を使うとアンフォーマットができることはわかりました。しかし、ただアンフォーマットを作るだけなら、トラック１周以上の長さのセクタ長でフォーマットしてやればいいのですから、あまりメリットはありません。実際、ＰＣ－９８００シリーズ本体だけでは、どちらの方法で作成したアンフォーマットも同じように見えます。

ただし、ハードウエアを追加していいなら区別は可能です。トラックの先頭部分が読めないのは確かですが、ＦＤＣはトラックの先頭をインデックスホールの信号によって認識しています。ですから、この信号を操作してトラックの先頭を変えてやればいいのです。例えば、次の方法が考えられます。

① インデックス信号を２回に１回間引く。

② インデックス信号を一定時間遅らす。

追加するハードウエアは、ドライブ装置とインターフェースボードの間にいれてやればよく、実現は容易です。この方法は、ハードウエアを追加した特定のＰＣでのみ動作するようにできるでしょう。

2.4 FDCの知識 ハングアップするってホント?

もう一つのμＰＣ７６５Ａの特殊な性質は、ある特定のデータ列が読み込まれるとハングアップすることです。ハングアップとはいつまでたってもＦＤＣが実行を終了しないことで、この状態になるとリセットをかける以外に止める方法はありません。

さて、それではどういう時にハングアップするのでしょうか。一般には、単密度で書かれたディスクを倍密度で読もうとすると、ハングアップすることがあるといわれています。

◎ハングアップの条件

調査の結果、ハングアップさせるための充分条件は、次の2つであることがわかりました。

① 倍密度で読んだときすべてのデータビットが0であり、それが永遠に続くこと。

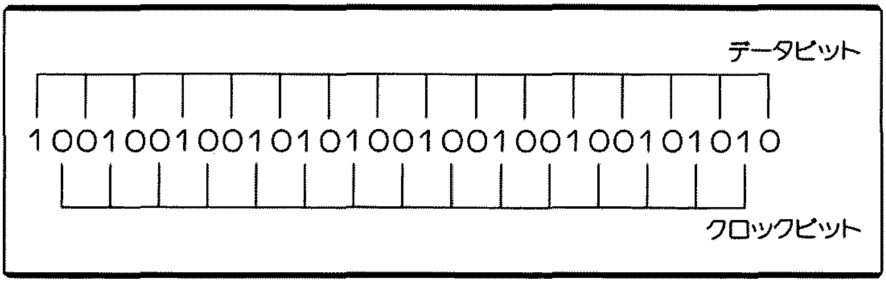
② 倍密度で読んだときクロックビットの並びの中に、111111010111111という並びが含まれていること。

これらの条件を満たしていればWrite IDを除くディスクにアクセスするコマンドなら、どれでもハングアップを起こします。

◎―――――――――――――― どうやって条件を満たすか

上の条件さえ満たせば、単密度に限らず倍密度でフォーマットしてもかまいません。しかし、任意の値を書き込めるＤＡＴＡ部分ならともかくマーク、ＧＡＰ、ＣＲＣなどの部分では、条件①はどうしても満たせません。例えば、ＧＡＰの４Ｅは次のようなデータ列になります。

図２．１１　ＧＡＰ４Ｅのデータビットとクロックビット

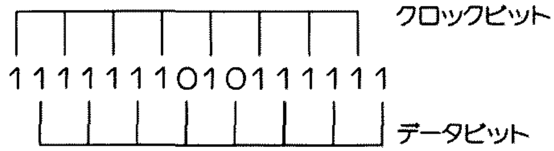


１つおきにデータビット、クロックビットがありますが、どちらをデータビットとみても必ず１が入っています。では、単密度でフォーマットしたらどうでしょうか。

単密度の書き込み周期は倍密度の２倍ですから、倍密度の周期で見れば単密度で書いたデータは、１つおきに必ず０になっているのです。ですから、単密度でフォーマットすれば条件①は自然に満たされ、条件②だけを考えればいいことになります。

つまり、単密度ではデータビット＋クロックビットで 111111010111111 を満たせばいいのです。単密度のクロックビットは通常すべて１ですから、先のデータ列は、次ページの図のようにデータビットとクロックビットに分けるのがいいでしょう。

図２．12　単密度で条件を満たすには



上がクロックビットで下がデータビットです。つまり、Ｅ７を書き込んでやればいいようです。実際にはこの１バイト分のデータが連続していますから、どこの点で区切って１バイトと読んでもいいわけで、データビットが２ビットだけ連続して０になるようなデータ（ＣＦ，９Ｆ，３Ｆ，７Ｅ，ＦＣ，Ｆ９，Ｆ３）なら、同じデータ列が得られます。以上から、ハングアップさせるためには、７Ｅ，ＣＦ，９Ｆ，３Ｆ，７Ｅ，ＦＣ，Ｆ９，Ｆ３などを初期値データとして単密度でフォーマットすればいいことがわかりました。

さて、これで必ずハングアップするかというとそうではありません。それは、トラックの先頭部分にデータの継ぎ目が生じますが、そこでビットずれを起こし条件①の読み込んだデータビットが、永遠に０であることを満たさなくなるからです。

データビットとクロックビットは１ビットおきに交互にあるわけですが、継ぎ目のところではデータビット同士、クロックビット同士が隣り合う可能性があるのです。もしそうだとすると、そこでクロックビットとデータビットが反転してしまい、データビットのつもりでクロックビットを読むことになってしまうからです。そうすると条件①の、永遠に０だけを読み続けることができなくなりハングアップせずに終了してしまいます。

◎ 何度かリトライする

結局、ハングアップするかどうかは継ぎ目にかかっていることがわかりました。しかし、何周してもデータビットだけを読み続けるようにうまく継ぎ目がつながるかどうかはまったくの偶然です。ドライブ装置の回転精度はそれほどいいものではありません。そこで、１度フォーマットしてリードを行ないハングアップしなければ、ハングアップするまで何度もフォーマットとリードを繰り返すという方法が考えられます。実験によると少なくとも１割以上の確率でハングアップは起きるようです。

◎ プロテクトへの可能性

このハングアップするという性質は次の２点でプロテクトに利用できそうです。

① コピーツールがコピーのためリードしようとした時点でＦＤＣがハングアップするため、コピーがそこで中断する。

専用ハードウエアを用いないコピーツールでは、フォーマット解析のため、ＦＤＣのRead ID，Read Diagnostic等のコマンドを使いますが、この処理が施してあれば倍密度でリードしようとする限り、必ずそこでハングアップします。これを有効に使うためには倍密度でリードさせなくてはいけませんが、コピーツールの方は、これに対処して始めに単密度でRead IDを行なうものが多いようです。この段階でＩＤが読めれば、倍密度でリードは行ないませんから、ＩＤが読めないようにアンフォーマットにしておくのがベストです。アンフォーマットにするには、フォーマット時に指定するセクタ長をトラック長より長くすればいいのです。

ただし、最近のコピーツールはハングアップした場合、ソフトウエアでＦＤＣにリセットをかけることによってこれに対処するものが現われています。注意して下さい。

②　ハングアップするかどうかをチェックする。

前の方法は、コピーを中断させるだけですが、この方法はアプリケーションの動作時に本物かどうかをチェックしようというものです。あるトラックが必ずハングアップするように処理をしておいて、アプリケーションの動作時にそのトラックをリードし、ちゃんとハングアップするかどうかをチェックするのです。ハングアップしてしまったらアプリケーションが続行できなくなるのでは、と思われるかも知れませんが、ソフトウエアでＦＤＣにリセットをかけることができますから大丈夫です。このプロテクトは強力で専用ハードウエアを使ってまるごとトラックをコピーしても継ぎ目において１ビットの誤差は出ますから、偽物はハングアップしなくなる可能性大です。

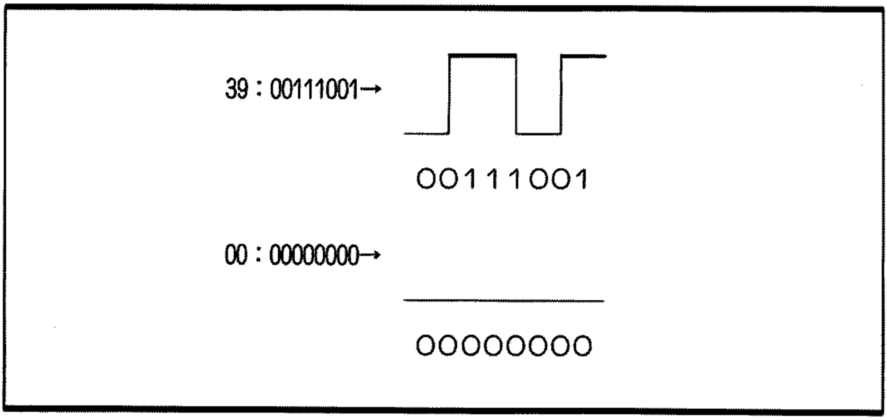


2.5 FDCの知識 データビットとクロックビット

◎ FMとMFM

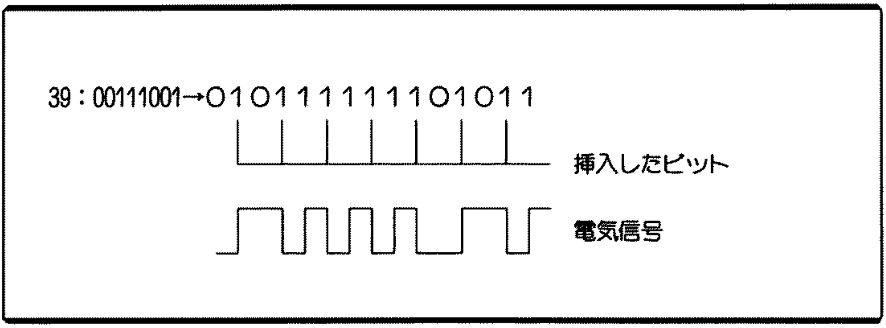
ここで、データビットとクロックビットについて説明しておきましょう。それには、まずディスクにはどうやって情報が記録されているかという基本から説明しなくてはなりません。といってもフォーマットの話ではありません。ここでは、もっと細かい話になります。例えば、39という16進で1バイトのデータがディスク上では、どのように記録されているかということです。最終的には電気信号になってヘッドへ送られていることは確かです。少し、考えてみましょう。いちばん簡単なのはデータの0 1がそのまま電気信号のマイナス、プラスになることです。例えば、39なら次のようになります。

図2．13　データのビットをそのまま電気信号にすると



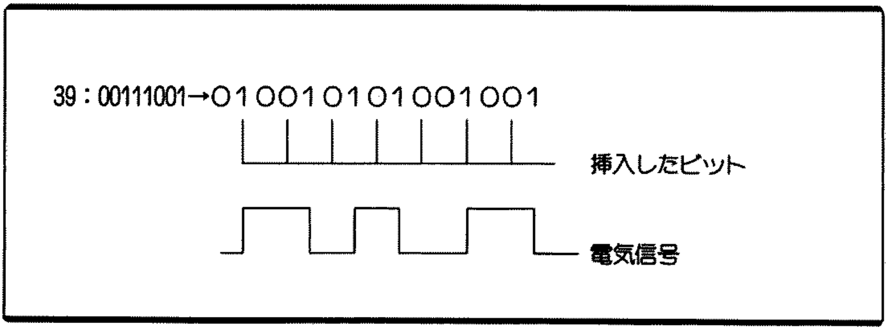
ところが、もし0ばかり、1ばかりが続くようなデータだと、ほとんど直流になってしまって、読んだときにうまくデータが再現されません。適当な間隔で0 1を繰り返すような信号にしないといけないのです。そこで、考えられたのが、FM方式と呼ばれるものです。まず、元のデータの各ビットの間にすべて1のビットを1つ挿入します。次にできた合成データ列の1のところで反転するように電気信号を作るのです。

図2．14　FM方式



この場合だと、どんなデータが入力されても電気信号の同じレベルが続くのは1ビットか2ビットの間だけです。ここで、1ビットおきに挿入したビットのことを**クロックビット**と呼びます。

図2．15　MFM方式



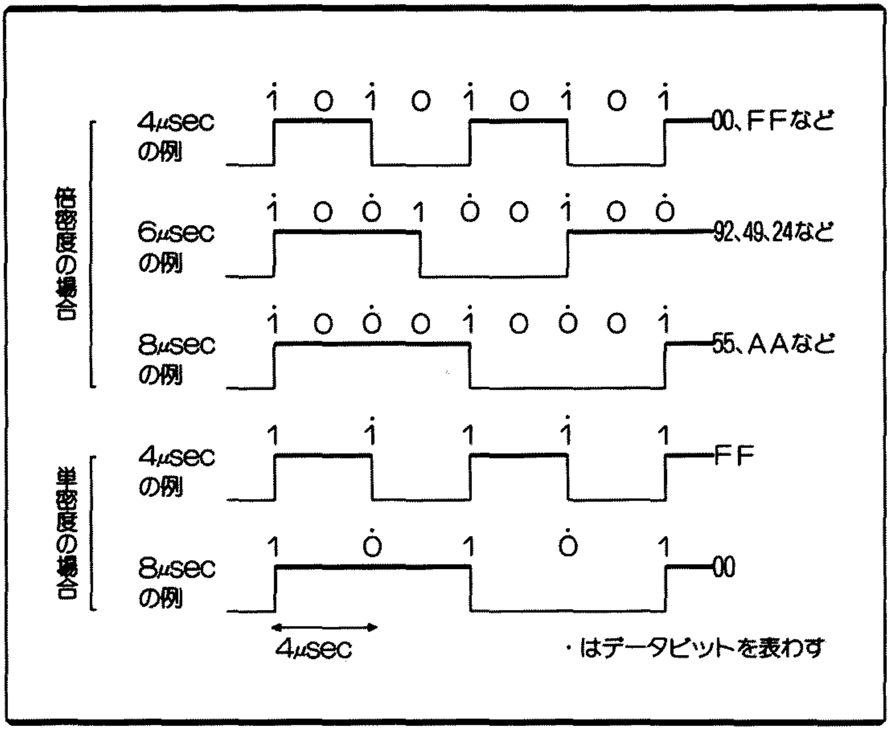
もう一つ有名な方式にMFM方式というのがあります。この方式も同じように挿入をするのですが、挿入するビットが常に1ではなく、挿入される場所によって1だったり、0だったりするというだけのことです。具体的には、挿入される場所で両隣りにある元のデータビットが両方とも0のときに限って1を、それ以外は0を挿入するという規則です。この方式では同じレベルが続くのは、2、3、4ビットの区間に限られます。よくFM方式、MFM方式というのを単密度、倍密度と同義に使われますが、そもそもの意味はここに示したように、挿入するクロックビットが異なっているだけです。

◎ 単密度と倍密度

単密度、倍密度という言葉は書き込めるデータの量を表わす言葉で、倍密度とは単密度に比べて同じ長さに2倍のデータを書き込めることを意味します。回転速度が変わらないとすると2倍の速さでデータを送り出しているのです。しかし、ディスクに書き込める信号の周波数にも限界があって、単純に2倍にはできません。もう一度、図2．14と図2．15を見てください。同じデータに対して作られる電気信号の周波数は、MFMがFMの2分の1になっているのがわかります。ということは、MFMの方は2倍の速さでデータを送り出しても、作られる電気信号の周波数はFMと同じになるだけです。このようにMFMは電気信号の周波数を変えずに2倍の速さでデータを送り出せる便利な方式なのです。単密度ではFM、倍密度ではMFMが用いられるのはこのためです。

それでは、単密度、倍密度の実際の電気信号はどうなっているでしょうか。2DDの場合倍密度ではデータ列の送り出される周波数は500kHz、つまり、ビットとビットの間は2μsec、電気信号の変化する間隔はその2、3、4倍、4、6または8μsecとなります。単密度では半分の250kHzで4μsec、電気信号の変化する間隔はその1または2倍ですから、4または8μsecとなります。範囲としては両者とも4～8μsecですが、単密度の方には6μsecという間隔はありません。

図2．16　電気信号の変化する間隔



◎―――――――――――― 単密度を倍密度で読むと

電気信号から見れば倍密度には4、6、8μsecの3種類の間隔があり、単密度は4、8の2種類ですから、単密度は倍密度に含まれるといえます。ですから、倍密度で書かれたものは必ずしも単密度として読めませんが、単密度で書かれたものは倍密度として読めるはずです。

それでは、一つ例を挙げてみましょう。単密度で書かれた3939というデータは倍密度では、どのように読めるでしょうか。まず、1ビットおきに1を入れて、合成データ列を求め電気信号を書いてみます。今度は逆に電気信号を2倍の周波

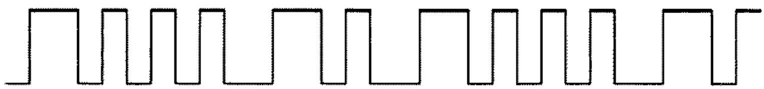
数で読み込みます。これを1ビットおきに読んだものが、倍密度で読んだときのデータとなるわけです。

図2．17　単密度を倍密度で読むと

3939:

0　0　1　1　1　0　0　1　0　0　1　1　1　0　0　1

0101111111010111010111111101011



0010001010101010101010001000101010001000101010101010100010001010

0101111111010111010111111101011
または
0000000000000000000000000000000

どこから読み始めるかで2通りあり、単密度の合成データ列と同じになるか、全部0になるかのどちらかです。このどちらになるかはわかりませんが、どちらかになることを理解しておいてください。このとき一方がデータビットで、もう一方がクロックビットに相当するわけですが、見てわかるように必ずしも倍密度のクロックビット生成規則（両隣りが0のときのみ1）に従ってはいません。しかし、読むときには、それは問題とならないようです。

◎――――――――――――ミッシングクロック

データビットとクロックビットについて、一応理解していただけたことと思います。ただし、一つだけ例外があって先のクロックビットの生成規則に従わない場合があります。それも1となるべきところが0となるので、これをミッシングクロックと呼びます。このミッシングクロックが使われるのはＩＭ，ＩＡＭ，Ｄ

ＡＭ，ＤＤＡＭだけで、これらのマークを他のコードと区別できるようにしているのです。つまり、倍密度のＤＡＭはＡ１，Ａ１，Ａ１，ＦＢですが、たとえ同じデータがＤＡＴＡ中にあってもまったくＤＡＭと認識されないのは、ミッシングクロックがないからです。



3.1 BIOSの知識
BIOSとは何だろう

◎ コマンド一覧

ＰＣ－９８００シリーズでは、ディスク入出力に必要な制御プログラムがサブルーチンとしてROM内に格納されています。これをＢＩＯＳと呼びます。ディスク入出力には、ＦＤＣ，ＤＭＡ，割込みなど煩雑なハードウエアの制御が必要ですが、ＢＩＯＳを利用すれば、少数のパラメータを与えるだけでよく、プログラムが非常に簡単になります。

ＢＩＯＳコマンドには、μＰＤ７６５Ａのコマンドにほぼ１対１に対応して、13種類のコマンドが用意されています。ＦＤＣにあってＢＩＯＳにまったく入っていないコマンドは、Scan Equal，Scan Low or Equal，Scan High or Equalの３つがありますが、これらは、プロテクトには必要ないので問題にはなりません。

プロテクトにおいて、ＢＩＯＳを使うことの問題点は、ＦＤＣに与えるパラメータのすべては指定できないことで、ＦＤＣに渡されるパラメータのいくつかは、ＢＩＯＳの中で決められてしまいます。このことは、標準的な使い方をする分には便利ですが、ハードウエアのもつ機能を100％引き出そうとすると障害になります。ただし、ＰＣ－９８０１Ｆ／ＶＦ／ＶＭ／ＸＡ／Ｕでは、それらのほとんどに対処可能です。その方法は後述します。

ＢＩＯＳコマンドの一覧を次ページに掲げます。

表３．１　ＢＩＯＳコマンド一覧

BIOSコマンド	解　　説	対応するＦＤＣコマンド
SEEK	ＢＩＯＳでは、すべてのコマンドがシーク機能を持つ。ですから、シークしかしないコマンドは何もしないコマンドである。（No Operation）	Seek
VERIFY	ベリファイといってもメモリのデータとディスクのデータを比較してくれるわけではない。ＦＤＣは普通にリードをするのだが、DMAがさかのぼってメモリに格納しないだけのことである。にも拘らずデータ領域を設定しなくてはならない。	Read
READ DIAGNOSTIC	重要なコマンド。2章ＦＤＣの知識を参照。	Read Diagnostic
INITIALIZE	ＢＩＯＳ、ＦＤＣの初期設定。普通は、既に設定されているので必要ないが、リセットをかけた後で必要になる。	Specify Recalibrate
SENCE	ライトプロテクト（書き込み禁止）の状態、または8インチでは、片面媒体か、両面媒体かを調べるのに使う。	Sence Device Status
WRITE	重要なコマンド。２章ＦＤＣの知識参照。	Write
READ	重要なコマンド。２章ＦＤＣの知識参照。	Read
RECALIBRATE	トラック０にヘッドを移動。エラーの原因がシークの失敗にあると思われるときに端から数え直すためにトラック０に戻すのに使う。	Recalibrate
WRITE DELETED DATA	重要なコマンド。２章ＦＤＣの知識参照。	Write Deleted Data

READ ID	重要なコマンド。２章ＦＤＣの知識参照。	Read
READ DELETED DATA	デイリーテッドデータを読むためのコマンドだが、普通のＲＥＡＤでも充分に読め、リターンコードでデイリーテッドデータであることもわかるので、このコマンドは特に必要ない。	Read Deleted Data
FORMAT	WRITE IDのこと。重要なコマンド。２章ＦＤＣの知識参照。	Write ID
SET OPERATION MODE	２ＤＤのドライブで２Ｄのディスクを読むときに使う。２ＤＤのドライブをユニット毎に、２Ｄに指定できる。読み込みはできるが、書き込みは保証されないので、２ＤＤのドライブで２Ｄのディスクを作成することは危険。	

3.2 BIOSの知識 BIOSにできないこと

コマンドの種類に関してはまったく問題ないのですが、ＦＤＣに与える一部のパラメータが固定されているため大きな制約を受けます。通常、ＢＩＯＳを使うと次のことができなくなります。

① Write ID（フォーマット）時にＧＡＰ３の長さを任意に設定すること。セクタ長（Ｎ）と記録密度から定まる標準値が常に使われます（表３．２）。

② Write ID（フォーマット）時にＳＣ（セクタ数）を任意に設定すること。セクタ長（Ｎ）と記録密度から定まる標準値が常に使われます（表３．２）。セクタ数を少なくする分には、ＢＸに与えるデータ長を短くすることで実現可能ですが、セクタ数を標準値より多くすることはできません。

③ 単密度で1024バイト以上、倍密度で128バイトまたは2048バイト以上のフォーマット。できないことはありませんが、ＧＡＰ３，ＳＣの値が用意されていません。

④ 書き込み中にＦＤＣにリセットをかけること。リセットはプロテクトでは重要なテクニックの１つですが、ＢＩＯＳは処理がすべて終了するまで呼び出し元に制御を移さず、書き込みの途中でリセットをかけることができません。タイマと割り込みを使えばできないことはありませんが精度が問題となります。

表3．2 WRITE IDで使われるGAP3，SCの標準値

(1) 2HDまたは8インチの場合

セクタ長コード N	セクタ長 (バイト)	倍密度		単密度	
		SC	GAP3	SC	GAP3
0	128	X	X	26	27
1	256	26	54	15	42
2	512	15	86	8	58
3	1024	8	116	X	X

SC：1トラックのセクタ数，X：未定義

(2) 5インチ2DDの場合

セクタ長コード N	セクタ長 (バイト)	倍密度		単密度	
		SC	GAP3	SC	GAP3
0	128	X	X	16	27
1	256	16	54	9	42
2	512	9	86	5	58
3	1024	5	116	X	X

SC：1トラックのセクタ数，X：未定義

3.3 BIOSの知識
こうすればBIOSでも使える

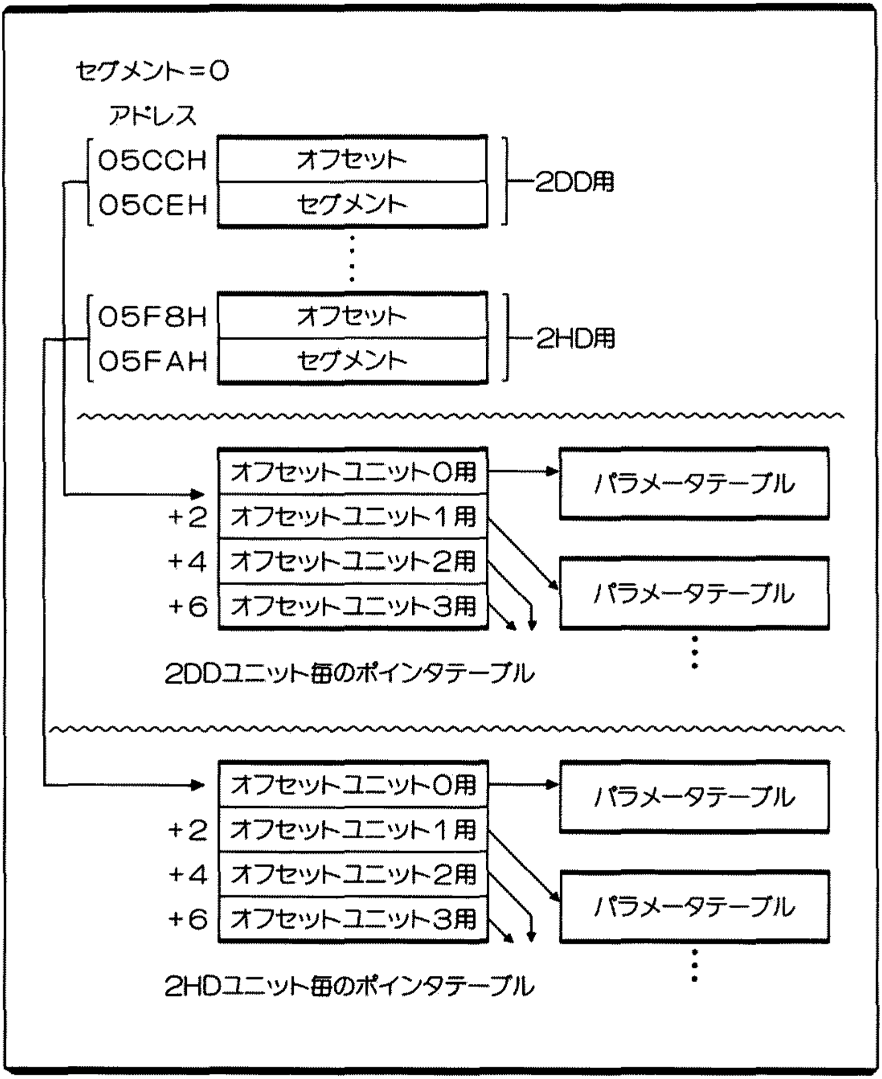
ＢＩＯＳの解析の結果機種は限られますが、通常、固定されていたパラメータもユーザが指定できることがわかりました。ＰＣ－９８０１Ｆ／ＶＦ／ＶＭ／ＸＡ／Ｕに限り、上の①，②，③は解決できます。

ＦＤＣに渡すＧＡＰ３，ＳＣなどのパラメータは、ＲＯＭ内にパラメータテーブルとして用意されていますが、そのテーブルを指す元となるポインタがＲＡＭ上（システム共通域）にあるので、そのポインタを書き変えれば、自分で用意したパラメータテーブルを使わせることができます。

ポインティングは２段階で行なわれています。パラメータテーブルはユニット毎に１つずつ用意することができ、各パラメータテーブルの先頭アドレスを示すユニット毎のポインタテーブルを用意します。ユニット毎のポインタテーブルは、１ユニットにつき２バイトでオフセットアドレスを表わしています。セグメントは指定できませんから、パラメータテーブルとユニット毎のポインタテーブルは、同一セグメントになくてはいけません。さらに、このユニット毎のポインタテーブルの先頭を指すポインタがあって、２ＤＤ用が［５ＣＣＨ－５ＣＦＨ］に、２ＨＤ用が［５Ｆ８Ｈ－５ＦＢＨ］にあります。セグメントはいずれも０です。先頭の２バイトがオフセット、後の２バイトがセグメントを表わしています。

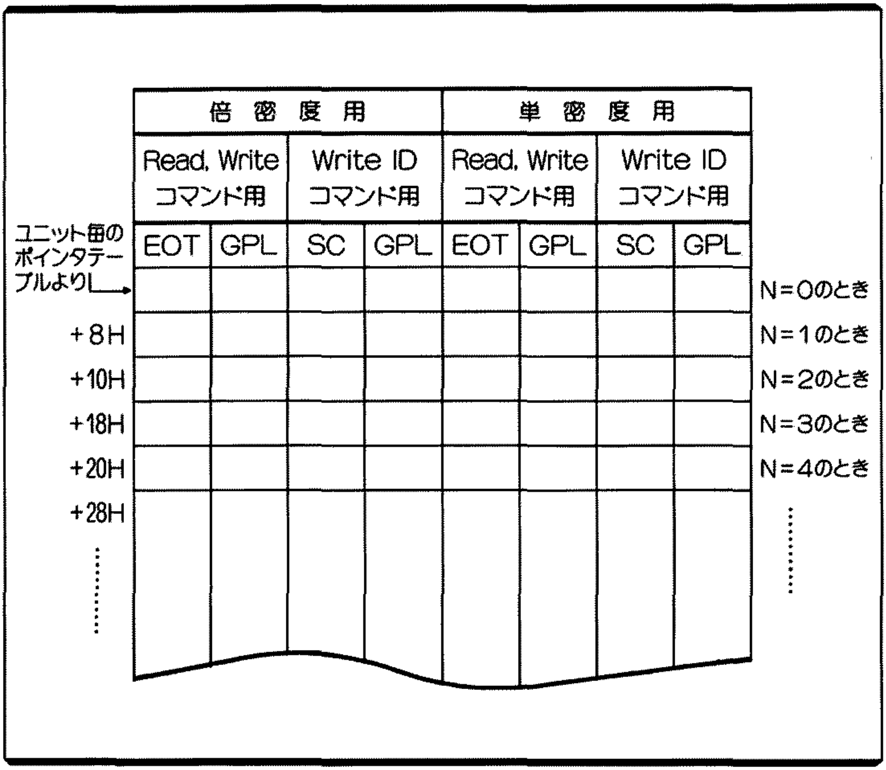
このようにユニット毎に異なったパラメータテーブルを用意できるようになっていますが、全ユニットが同じパラメータでよいときには共通で１つのパラメータテーブルを使えばよく、ＲＯＭでは各ユニットのポインタは、同じパラメータテーブルを指しています。

図３．１　ポインティングの構造



パラメータテーブルの構造を次に示します。

図３．２　パラメータテーブルの構造



ＰＣ−９８０１／Ｅ／Ｍにはこのようなポインタがありませんので、自分でＢＩＯＳを作らねばなりません。本書では付録に、プロテクトに必要な機能を備えた拡張ＢＩＯＳを掲載しました。御利用ください。

ＰＣ−９８０１ＶＭ／ＸＡでは、２ＤＤ、２ＨＤ両方のタイプのディスケットを使うことができますが、２ＤＤとして使うときは２ＤＤ用の、２ＨＤとして使うときは２ＨＤ用のポインタが使われています。

リセットに関しては、やはりＢＩＯＳの作り直しが必要です。付録の拡張ＢＩＯＳを利用してください。

3.4 BIOSの知識 BIOSの基本的な使い方

◎ ＢＩＯＳコール手順

ＢＩＯＳは機械語命令ＩＮＴ　１ＢＨでコールします。どのコマンドかはＡＨレジスタの内容で決まります。必要なパラメータ等もレジスタに入れて渡します。書き込むデータはメモリ上に用意し、先頭アドレスをレジスタに入れて渡します。読み込みの場合も、読み込んだデータはメモリ上に格納されますので、その先頭アドレスをレジスタに入れて渡します。

ＩＮＴ　１ＢＨでＢＩＯＳが呼び出されコマンドに応じて処理されます。結果はリターンコードとしてＡＨレジスタに返されます。また、キャリーフラグはエラーのときセットされます。Read ID コマンドに限って読み出されたＩＤ情報がレジスタに格納されて戻ります。

ＢＩＯＳコールの一般的な手順は次のようになります。

１．各種パラメータをレジスタにセットする。

　　ＭＯＶ　ｒｅｇ，××××等

２．必要に応じメモリ上のデータ領域の先頭アドレスをレジスタにセットする。

　　ＭＯＶ　ＢＰ，××××　　（オフセット）

　　ＭＯＶ　ＥＳ，××　　　（セグメント）

３．ＩＮＴ　１ＢＨ

４．エラー時の処理のため分岐

　　ＪＢ　　ＥＲＲＯＲ等

◎ 入　力

次の表は各コマンドで入力に使われるレジスタを示しています。

表3．3　各コマンドで入力に使われるレジスタ

BIOS コマンド名	No.	入力								
		AH	AL	BX	CH	CL	DH	DL	ES	BP
シーク	0	①	③			⑦				
ベリファイ	1	①	③	⑤	⑥	⑦	⑧	⑨	⑪	⑫
リードダイアグ	2	①	③	⑤	⑥	⑦	⑧		⑪	⑫
イニシャライズ	3	①	③							
センスステータス	4	①	③							
ライトデータ	5	①	③	⑤	⑥	⑦	⑧	⑨	⑪	⑫
リードデータ	6	①	③	⑤	⑥	⑦	⑧	⑨	⑪	⑫
リキャリブレイト	7	①	③							
シーク	8	①	③			⑦				
ライトデイリーティッドデータ	9	①	③	⑤	⑥	⑦	⑧	⑨	⑪	⑫
リードID	10	①	③			⑦	⑧			
シーク	11	①	③			⑦				
リードデイリーティッドデータ	12	①	③	⑤	⑥	⑦	⑧	⑨	⑪	⑫
フォーマット	13	①	③	⑤	⑥	⑦	⑧	⑩	⑪	⑫
セットオペレーションモード	14	②	④							
シーク	15	①	③			⑦				

表中、番号のあるレジスタはその番号のところで意味を解説しています。空白のところはそのレジスタが使われないことを意味します。表を見てわかるようにコマンドは異なっても、同じレジスタは同じ使われ方をすることが多く、覚えや

すくなっています。次に各レジスタの意味を示します。

シークコマンドが4箇所にありますが、実はこれらは何もせずにリターンするコマンドなのです。しかし、シーク機能だけはすべてのコマンドが持っているので、あえて言えばシークということです。レジスタＡＨのＳＫビットを０にすれば、それこそ何もせずにリターンします。

① ＡＨ　　　コマンドNo.その他

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
MT	MF	r	SK	コマンドNo			

ＭＴ：マルチトラック機能を使うか

０：使わない　　１：使う

この機能を使うとヘッド０の側からヘッド１の側へ連続的に読むことと書くことができます。

ＭＦ：単密度か倍密度か

０：単密度　　１：倍密度

ｒ：エラーのときリトライを許すか

０：許す　　１：許さない

許すにするとリターンコード５０Ｈ以上のエラーが発生したとき、８回まで自動的にリトライ（再試行）します。プロテクトではわざとエラーが発生する条件で使用したり、タイミング、実行時間が問題となることも多くリトライは許さないようにして使うことが多いようです。リトライが必要なときにはコールする側で繰り返すようにします。

ＳＫ：シークするか

０：しない　　１：する

１にするとＣＬレジスタの値をシリンダ番号としてシーク（ヘッドをそのシリンダに移動）します。ＣＬレジスタは同時にＩＤのＣとのマッチングにも使われますから、物理的なシリンダ番号とＩＤに書かれ

ているシリンダ番号が異なるときは、このビットを0にしなくてはいけません。

コマンドNo.：各コマンドのNo.（表参照）を指定します。

③ AL　　デバイス種別、ヘッド反転フラグ、ユニット番号

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
デバイス種別					HR	UNIT	

デバイス種別：8インチ、5インチ2HDか、2DDか

7：5インチ2DD　　9：8インチまたは5インチ2HD

HR：ヘッド反転フラグ　物理的なヘッド番号は原則としてDHレジスタの最下位ビットが使われるが、このビットを1とするとその反対側のヘッドにアクセスできる。

0：DHレジスタのとおり　1：DHレジスタの反対

UNIT：ユニット番号　0～3のユニット番号を指定する

②および④　　セットオペレーションモード

セットオペレーションモードコマンドでは、AH，ALの意味が異なります。このコマンドはデバイスが2DDのときのみ有効で、ユニットごとにディスケットのトラック密度、両面／片面などの設定を行なうものです。両方ともワークエリアの設定を行なうだけで、ハード的には何も変更されません。特に両面／片面は、BIOSでは使われないので全く意味を持ちません。

AH：トラック密度の設定か両面／片面の設定の切り換えとコマンド番号

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
TS			SK	コマンドNo.			

ＴＳ：トラック密度の設定をするか両面／片面の設定をするか
0：両面／片面の設定　1：トラック密度の設定
その他は①に同じ

ＡＬ：デバイス種別およびユニット毎の設定

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
デバイス種別				U3	U2	U1	U0

デバイス種別：②に同じ、ただし、２ＤＤの場合のみ意味を持つ。

Ｕ０～Ｕ３：対応するユニットをＴＳに従って設定する
ＴＳが０のとき
0：片面　　1：両面
ＴＳが１のとき
0：40トラック　1：80トラック

トラック密度が40トラックとは80トラックの偶数番目だけにトラックがあることに等しい。そこで、２ＤＤのドライブでも、１つおきに読めば２Ｄのディスクが読めるのである。ＢＩＯＳではここで０（40トラック）に設定されたユニットでシークを行なうときにはシリンダ番号を自動的に２倍することによってこれを実現している。これで、読み込みは可能だが、２Ｄと２ＤＤではそもそもヘッドの幅が違うので、２ＤＤのドライブで書いたディスクを２Ｄのドライブで読ませるのは危険である。

⑤　ＢＸ　　転送するデータのバイト数（16ビット）

⑥　ＣＨ　　セクタ長コード
コードとバイト数の関係は表２．５に示すとおり。０～255のいずれ

を指定してもいいが、8以上はすべて32768バイトとなる。

⑦ CL　　シリンダ番号

物理的シリンダ番号とソフト的シリンダ番号（IDに書かれているC）の2つの意味があり、その両方に使われる。両方が一致しているときは問題ないが、異なるときは工夫が必要。まず、シークだけを行なうコマンドで物理的シリンダを合わせ、次に行なうリード、ライトではAHのSKビットを0にしてコマンドを出せばよい。

⑧ DH　　ヘッド番号

シリンダ番号と同じく、物理的ヘッド番号とソフト的ヘッド番号（IDに書かれているH）の2つの意味がありその両方に使われる。物理的ヘッド番号は0か1しかないので、DHの最下位ビットのみが相当する。つまり、偶数ならヘッド0、奇数ならヘッド1となる。その反対側にアクセスするには、③で説明したHRを1にすれば可能である。

⑨ DL　　セクタ番号

IDに書かれているRとのマッチングに使われる。

⑩ DL　　初期値データ

Write IDでDATA部を埋める1バイトのデータ。DATA部はすべてこの1バイトを繰り返して埋められる。

⑪ ES　　バッファ領域先頭のセグメント

データ転送を伴うコマンドではメモリ上にバッファ領域を確保するが、その先頭アドレスのセグメントを指定する。DMACの制約からバッファ領域は、64キロバイトバウンダリをまたがないように確保しなくてはならない。つまり、20ビットアドレスの上位4ビットが領域内で変わってはならない。

表３．４　リターンコードの意味

コード	意　　味
00H	正常終了
10H	読み込んだデータがディリーティッドデータであった。これは、特にエラーではない。
20H	バッファ領域が64キロバイトバウンダリをまたがっている。DMAの設定エラーが生じたもので、ＦＤＣには全くコマンドは送られない。
30H	マルチセクタ機能を利用したとき、セクタ番号がＥＯＴに一致してもなおデータ転送を続行しようとした。リードダイアグでは、ＥＯＴ個以上のセクタを読もうとした。エラーではあるが、セクタ番号ＥＯＴまでは正しく処理されている。
40H	普通、このエラーは出ません。もし、出るならディスクドライブ装置の故障です。
50H	データ転送のスピードがディスクの回転に追いつかなかったことを意味しますが、DMAを使っている限りこのエラーは出ないはずです。
60H	ディスクドライブがNOT READY
70H	ライト系のコマンドで書き込み禁止となっている。勿論、書き込みは行われません。プロテクトノッチの状態を調べることができる。
A0H	ＩＤのＣＲＣ検査でエラーが発生。データ処理は行われません。
B0H	ＤＡＴＡ部のＣＲＣ検査でエラーが発生した。一応、データは読み込まれている。ＩＤのＮをフォーマット時のセクタ長と異なるものにするとこのエラーとなる。
C0H	ＩＡＭは見つかったが指定したＩＤと一致するＩＤを持ったセクタが見つからなかった。データの処理は行われません。
E0H	トラックを１周してもＩＡＭが見つからなかった。異なる記録密度で読もうとしたり、アンフォーマットのトラックを読もうとするとこのエラーになります。
F0H	IDまでは見つかったが、ＤＡＭまたはＤＤＡＭが見つからなかった。

⑫ ＢＰ　バッファ領域先頭のオフセット

◎――出　力

Read ID コマンドでは、読み込んだＩＤ情報をレジスタに入れて返します。レジスタと意味は次のとおりです。

ＣＬ：ＩＤのＣ
ＤＨ：ＩＤのＨ
ＤＬ：ＩＤのＲ
ＣＨ：ＩＤのＮ

◎――リターンコード

すべてのコマンドでＡＨレジスタにリターンコードが返されます。リターンコードはコマンドの実行でエラーが発生した場合、そのエラーの内容を示します。リターンコードの値と意味は前ページの表のとおりです。

4.1 プロテクトの知識 プロテクトをかけるには

２章、３章を通してプロテクトに必要な予備知識は修得されたことと思います。本章では、それらを基にしてプロテクト理論を完成させたいと思います。

１章でも触れたように、コピーできないのは、

① ＰＣ－９８００以外の特殊なハードウエアを用いてディスケットを作成した場合。

② ＰＣ－９８００を用いて作成したものであるがその手順がわからない場合。

③ ＰＣ－９８００を用いて作成したものでありその手順もわかったが、同じ手順を用いても誤差のために同じものが作成できない場合。

の３つの場合があります。それでは、それぞれの場合を検討してみましょう。

◎―――――――― 特殊なハードウエアで作成した場合

ＰＣ－９８００で実現できないフォーマットを作れば、ＰＣ－９８００ではコピーできなくなるのは明らかです。「ＰＣで作れないものがＰＣで読めるのか」などという疑問を持つ人はもういないと思いますが、もし、あなたが疑問に感じるのなら２章に戻って読み直したほうがよさそうです。この場合でも、似たようなハードウエアを使えばコピーできるわけで、現実にそういったハードウエアが出回わっている現在、やはり、手順、誤差などが問題となってきます。

◎ 手順がわからない場合

１枚のディスケットがバージンの状態から製品として完成するまでには、何回かディスクには書き込みが行なわれるでしょう。それがどういう手順で行なわれるかということです。ディスクに対して書き込みが行なえるのは、Write ID，Write Data，Write Deleted Data の３つのコマンドのみです。これらのコマンドをどういう順番で、どういうタイミングで、どういうパラメータとともに、また、どういうデータとともに与えたか、それでそのディスクは決定されます。コピーツールの仕事はこれを推測し、同じ手順を実現することです。もっとも、手順が違っても、できたものが同じであればそれでかまいませんが。強いプロテクトとは、この推測することが難しいものといえます。

◎ 誤差のために同じものができない場合

何にでも誤差はつきものです。特に、フロッピディスクのように機械的機構の多いものでは、かなりの誤差が生じます。特に顕著なのが回転精度の悪さから生ずる誤差で、部分的にはビットずれとして、全体的にはトラック容量の違いとして観測されます。

4.2 プロテクトの知識 作成手順をわからなくするには

◎ コマンドの順番

手順を決定する要素として、コマンドの順番、タイミング、パラメータ、データの4つをあげました。まず、それぞれを検討してみます。

コマンドはWrite ID，Write Data，Write Deleted Dataの3つです。これらのコマンドを与える順番としては、1番目にWrite ID、2番目以後Write DataまたはWrite Deleted Dataの繰り返しというのが普通です。これは、Write IDコマンドは必ずトラック1周すべてに書き込みを行なうので、Write IDをたとえ複数回行なっても最後の1回分しか結果に残らないからです。ところが、リセット（後述）を用いることを考慮に入れるとこの前提は大きく崩れます。Write IDの途中でリセットをかけることによって、トラックの一部にしか書き込みをさせないことができるからです。何回か前のWrite IDの結果を残すことも可能なのです。結局、リセットによってコマンドの順番も非常に多くの可能性が出てきます。

◎ タイミング

ＦＤＣにいつコマンドを与えるかです。Write ID コマンドに関していえば、いつコマンドを与えても結果は同じです。それは、このコマンドが必ずトラックの先頭（インデックスホールの位置）から書き始めるからです。ところが、Write Data などのコマンドでは、コマンドを与えた時点にヘッドのある位置

からＩＤを探し始め、最初にＩＤが一致したセクタに書き込みを行ないますから、同じＩＤが複数ある場合には、タイミングによって異なるセクタに書き込みが行なわれます。

さらに、リセットを用いる場合は任意のタイミングでリセットをかけることができ、その時点でＦＤＣはただちに書き込みを中止しますから、それがそのまま結果に反映されます。コマンドのタイミングはせいぜいセクタ数ぐらいまでしか種類がありませんが、リセットのタイミングはビット単位として数万種類はあります。

また、データ転送の途中でＦＤＣのＴＣ入力をＯＮにすることも考えられますが、この場合は以後のデータが 00 になるだけで、データを変えることで同じ結果となるため、考慮する必要はありません。

◎ パラメータ

今度は一つひとつのコマンドについて検討します。コマンドは、パラメータとデータによって決まります。Write (Deleted) Data コマンドにもパラメータはありますが、結果に影響を与えないので、ここでは、Write ID コマンドについて考えます。Write ID コマンドには結果に影響を与えるパラメータは 4 つあります。Ｎ、ＳＣ、ＧＰＬ、Ｄです。Ｎはフォーマット時のセクタ長を決め、ＳＣはセクタ数を、ＧＰＬはＧＡＰ３の長さを、ＤはＤＡＴＡに書き込まれる値を決めます。これらのうちＤだけは後から書き込むデータによって変更可能ですが、その他の 3 つは、原則として後から変えることはできません。古典的なプロテクトでは、この 3 つのパラメータのうちいくつかを標準的でない値にしたものが多いようです。各パラメータは 0 ～255 の値をとることができますから、かなりバラエティはあります。ただし、Ｎは 6 または 7 以上でアンフォーマットになるため、それ以上では区別がしにくく、さらに 8 以上ではセクタ長は 32768 に固定されるので変化しません。ＳＣも一定以上に大きくなると書き込みが 1 周で収まらなくなり、2 周以上書き込みをしたり、条件によっては 1 周を少しはみだ

して書き込みを終わったりします。２周以上した場合、結果は最後の１周分しか残らないのは確かですが何周したかによってセクタの絶対的な位置が変わってきますので、検出不可能ではありません。

◎ データ

Write ID では、ＩＤに書き込むデータ、Write (Deleted) Data では、ＤＡＴＡに書き込むデータのことです。

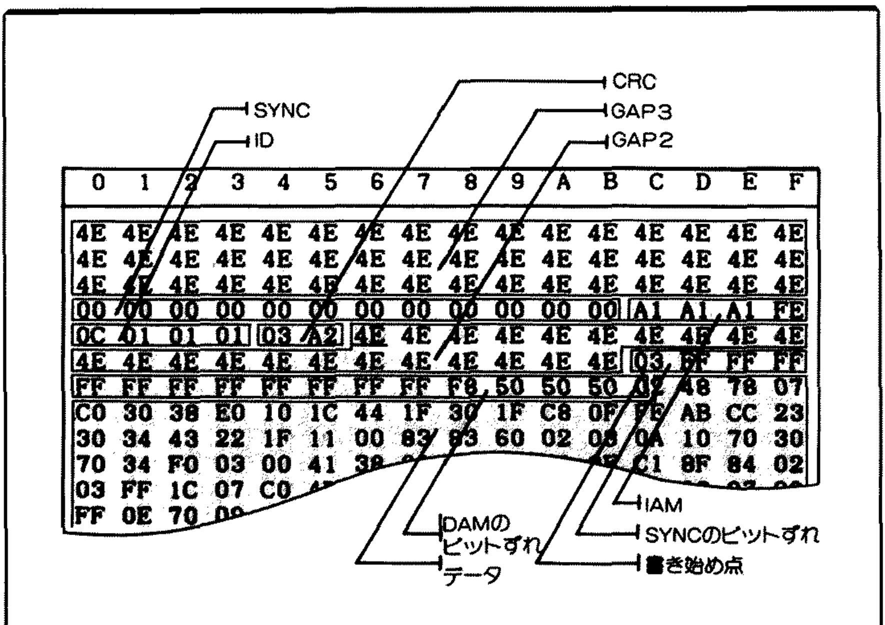
このように，以上説明したうちの何１つ変えても結果は違ってきます。ですから、プロテクトとはほとんどの場合、それが期待どおりかどうかチェックすることです。もちろん場合によっては、同じ結果を得るのに複数の方法があることもありますが、同じ結果を得るのにその手順が１つしかなければ、作成した手順とまったく同じ方法を用いなくてはなりません。

強いプロテクトとは作成手順が普通ではなく、結果から、作成手順がわかりにくく、かつ同じ結果が、それ以外の方法では得られないものということができます。上の４要素をもう一度再検討してください。

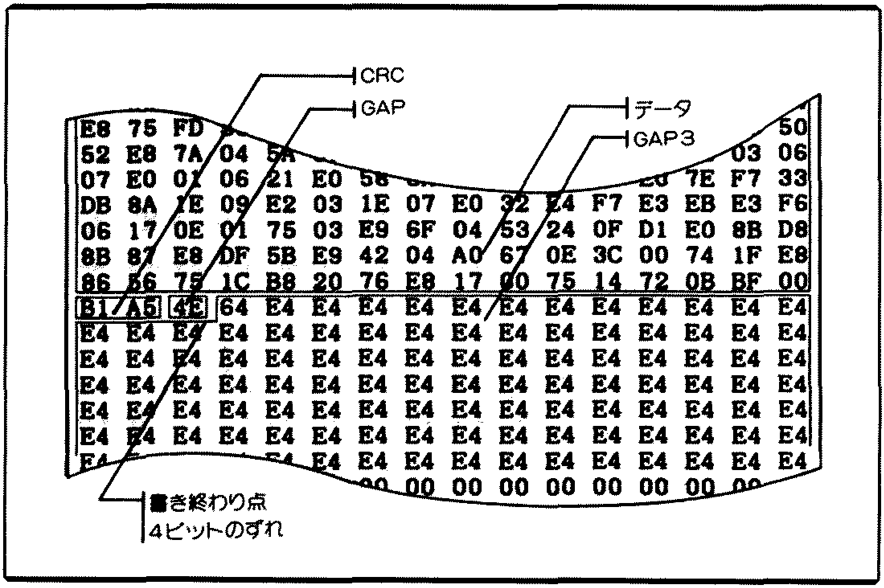
4.3 プロテクトの知識 ビットずれのいろいろ

先に述べた誤差の一つはビットずれとして観測されます。ビットずれとは、Writeコマンドなどで、1トラックの一部を書き直したとき、元に書いてあったものとピッタリ重ならず、その書き始めと書き終わりの点でずれることをいいます。　このビットずれは、ずれの度合いによっていろいろと観測されます。

図4．1　書き始め点でのビットずれ



書き終わり点でのビットずれ



ビットずれは大きくわけて観測されるデータが安定している、つまり、何度読んでも常に同じデータが読み込まれる場合と不安定な場合、つまり、読み取られるデータがときどき変化する場合があります。以下の説明では、書き重ねた部分が元の記録とピッタリ重なった場合を基準として、そこから何ビットずれているかということで場合わけします。また、ここでいう1ビットとは、データビットが書き込まれる間隔を意味しています。また、以下でNは整数を表わしています。

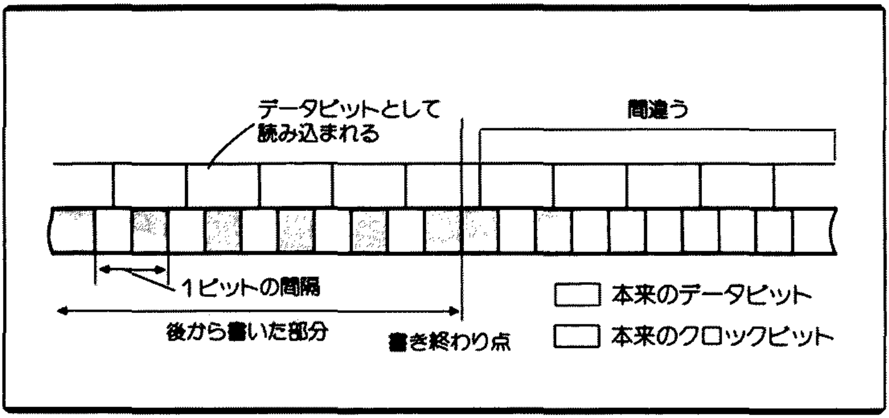
◎ 安定したビットずれ

ちょうど0.5ビット単位でずれたときには安定します。データビットの間にはクロックビットがはさまれていますから、0.5ビットという間隔はデータビットとクロックビットとの間隔です。

① 0．5＋Nビットのずれ（0.5，1.5，2.5，3.5など）

データビットで0.5ビットずれると、データビットの間には互いにクロックビットが入っていますから、クロックビットをデータビットと間違って読んでしまいます。クロックビットのパタンはデータビットとまったく異なりますから、この0.5ビットのビットずれが生ずると、それ以降はまったく違うパタンとなってしまいます。クロックビットがどうなるかは、2.5 クロックビットとデータビットを参照してください。

図4．2　0.5＋Nビットのずれ



② Nビットのずれ　(1，2，3，4，5など)
データビットがNビットシフトした形となります。

◎ 不安定なビットずれ

③ 0.25+0.5×Nビットのずれ
この場合は、ずれによってデータビットとクロックビットの中間点を読むようになります。といっても実際はどちらか一方です。しかし、安定していないので読むたびにデータビットであったり、クロックビットであったりします。このずれではビットずれ以降のデータが安定していません。

図4．3　Nビットのずれ

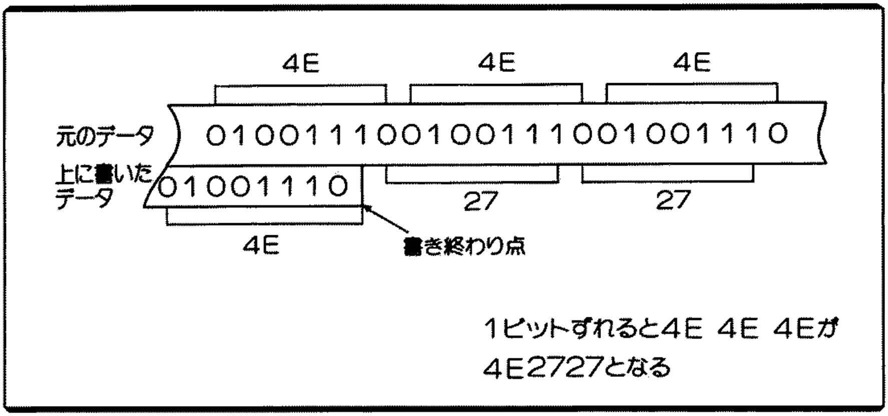
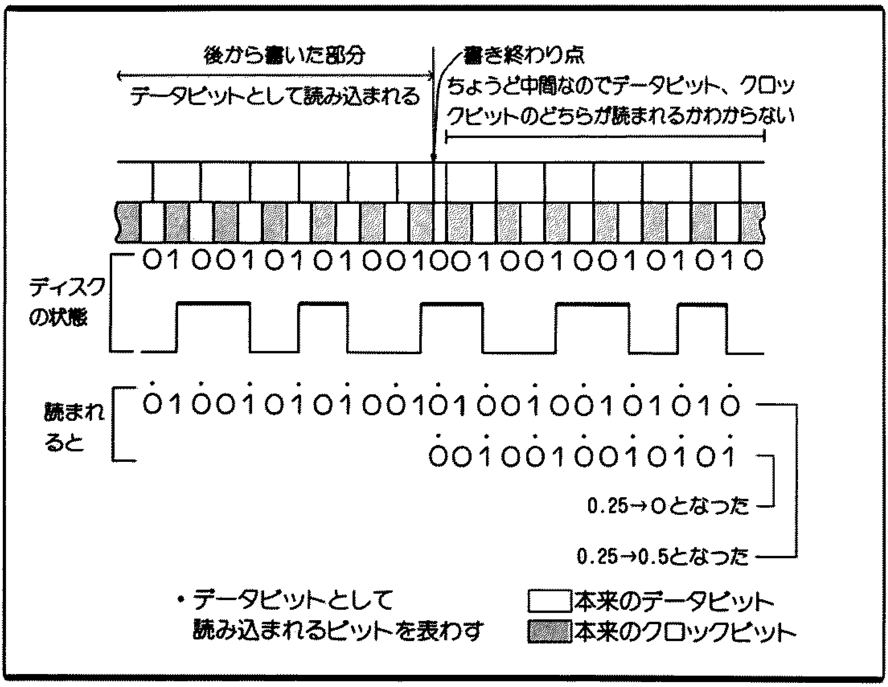


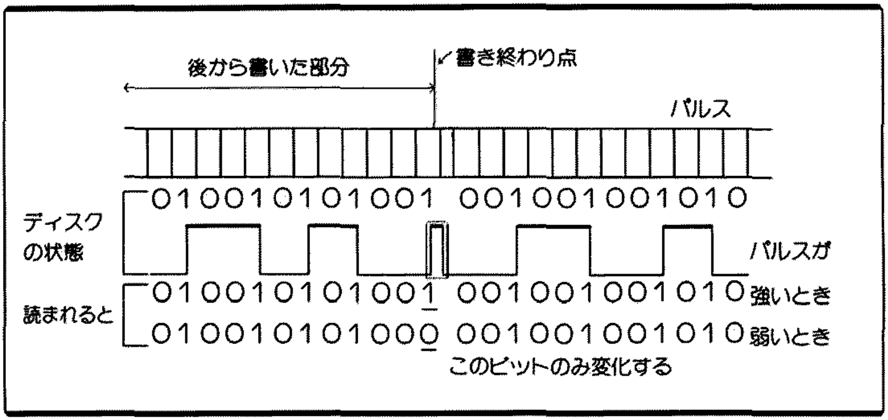
図4．4　0.25+0.5×Nビットのずれ



④　±0.1+0.5×Nビットのずれ

これは±0.1のずれがあると、場合によってその部分に短いパルスができる可能性があります。信号の変化点が1となりますからそこにパルスが検出されればそこのビットは1となりますが、検出されなければ0です。どちらになるかは読むたびに変わる可能性があり不安定です。ただし、この場合はビットずれの箇所のみで、以降には不安定な影響を与えません。

図4．5　±0.1+0.5×Nビットのずれ



このようにビットずれは、ずれの度合いによって4種類の場合が考えられます。逆にいえば、これだけの場合を想定すれば充分ということです。1度書き込みを行なえば書き終わり点では、このうちのいずれかが必ず起こっているわけです。ただし、このうちのどれが起こるかはまったく偶然です。もっとも書き込みを行なわなければ必ず0となりますが。プロテクトへの利用としてはこの偶然に発生したビットずれを観測しておきまったく同じビットずれを起こすかを、チェックする方法が考えられますが、③のように不安定な場合があり少々工夫が必要となります。詳しくは応用編を御覧ください。

4.4 プロテクトの知識 回転数とトラック容量

フロッピディスクは、1秒間に何回転するか知っていますか。5インチ2DDでは5回転、8インチと5インチ2HDでは少し速くて6回転です。

それでは、1トラックには何バイトのデータを書けるか知っていますか。これは、データがFDCから送られてくる速さがわかればわかります。データを送り出す周波数は、倍密度の場合2DDで500KHz、2HDまたは8インチで、1MHzです。これらをそれぞれの回転数で割って1トラックには2DDで100Kビット、2HD、8インチで167Kビットとなります。送り出されるデータの中にはデータビットとクロックビットがありますから、さらに2で割って50000ビットと 83500ビットとなります。バイトにすれば6250バイトと10437.5バイトとなります。これが1トラックに書ける容量です。単密度では、この半分になります。

このように、フロッピディスクのトラック容量はドライブ装置の回転速度によって決まるわけですから回転速度に誤差があるならば、当然トラック容量にも誤差が生じてきます。回転が遅くなるとトラック容量が増えます。

ドライブ装置の回転にはかなりの誤差があることはよく知られています。もっとも大きいのはドライブ装置の個体差、「ばらつき」で、規格では±2%以内となっているだけですから、それほど厳密には調整されていないようです。例えば、筆者の使用しているPC－9801Fのドライブを測定してみたところ、1周がドライブ1は 6304バイト、ドライブ2は 6345バイトでした。標準が 6250バイトですからそれぞれ+0.9%,+1.5%ということになります。さらに同じドライブ装置でも書くたびにすこしずつ変化しており、数分間に最大20バイト程の

ばらつきがありました。これは約0.3％です。みなさんも自分のドライブ装置について測定してみるといいでしょう。測定用のプログラムは付録2にあります。

この回転誤差はプロテクトへの利用もいろいろ考えられますが、最も単純で直接的なのが1周のバイト数を計ることです。1周に書き込まれるバイト数はフォーマットしたときの回転速度で決まります。フォーマットしたときとまったく同じ回転速度にしない限り、同じバイト数にはできないので、これは強力なプロテクトになります。ドライブ装置自体には回転速度を微調整するボリュームがついていますが、普通は手の届かない位置にあるので、ドライブ装置をはずさないことには調整できません。偶然の変動に期待するという手もありますが、一般に、偶然に変動する誤差よりも個体差の方が大きいので、たまたま最初からほとんど同じでない限り無理です。

このように回転変動を利用したプロテクトは、プロテクトの場所、方法がわかっても同じものが作れないという恐ろしいものです。

「そんなに回転変動があって、正しく読めるのか」と心配される方があるかも知れませんが、まったく心配はいりません。フォーマット時は書き込み周波数が一定のため、回転変動がそのままトラック容量の差となって現われますが、読み込み時はＶＦＯ（可変周波数発振機）により、読み込み周波数を変化させて回転に同期させているからです。実験によれば＋20％、1周に約7700バイト書き込んでも正常に読み出せました。

回転速度を変える方法も考えられないわけではありません。ドライブ装置のボリュームがいちばんですが、それを使えない場合は、

① 本体のフタをあけて、回転中にディスケットのジャケットを上から手で押さえる。摩擦が大きくなって回転速度が落ちます。

② 2ＤＤなら、モータのＯＮ／ＯＦＦを利用する。タイミングによって回転速度が落ちます。

③ 回転速度を変える代わりに書き込み周波数を変える。これにはハードウエアがいります。

4.5 プロテクトの知識 リセットを使ってみると

リセットはそもそも、2章で説明したハングアップに陥ったときにそこから抜け出すために設けられたものです。ところが、このリセットをライト系のコマンド実行中に使うと、その時点で書き込みが中止され、普通では実現できない状態をディスク上に残すことができます。つまり、4．2でも述べたように、リセットを積極的にプロテクトテクニックとして使うことによって、ディスク作成の手順をいくらでも複雑にすることができます。

リセットをかけるタイミングがディスクの状態を変える要因となり、また、そのタイミングを調べること、それを再現することが難しいために、有力なプロテクトテクニックとなるのです。

具体的な例は応用編で詳しく説明します。

4.6 プロテクトの知識 プロテクト組み込み上の注意

作成手順に工夫を凝らしたプロテクトの場合、フォーマットが異常になることもあり、異常な部分をOSから読み出し、書き込みするようなことのないように注意してください。OSの管理下からはずすにはディレクトリ、FATを操作して、その部分をシステム予約などの領域としてアクセスしないようにします。

次に各OSごとに実現方法を説明します。

◎ N88－BASICの場合

N88－BASICでは、クラスタごとにFATにどこが使用されているかが示されています。そこで、プロテクトをかけた部分のクラスタをFE（システム予約）とします。これでこの部分は使用されなくなります。

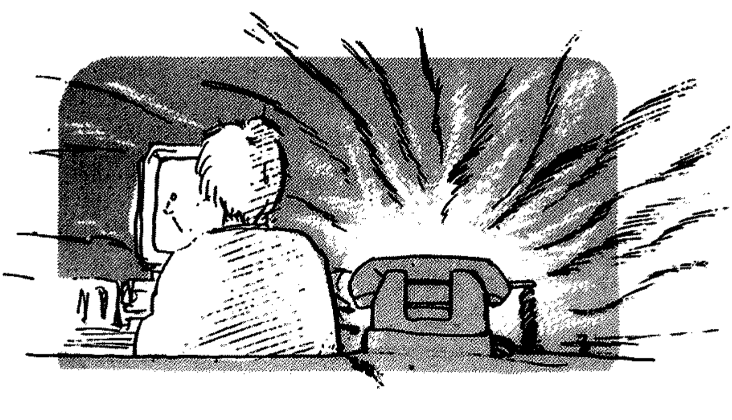
◎ MS－DOSの場合

MS－DOSでもFATによってどこが使用されているかが管理されています。MS－DOSでは、1.5バイト、12ビットで1クラスタを表わしていますが、値をFF7（欠陥クラスタを意味する）にすることによってそのクラスタにはアクセスされなくなります。

◎ CP／M－８６の場合

CP／M－８６ではFATがないので、ダミーのファイルを１つ作ってその領域をプロテクトに利用します。もちろん、そのファイルを読んだり書いたりすることのないようにしなくてはなりません。ファイル名をオールスペースなど見えないようにするといいでしょう。

また、普通使われていない内側のトラック（オーバートラック）を、プロテクトに用いれば上のような心配はいりません。



PC-9800 SERIES THE PROTECT

II 応用編

- ○基本テクニック
- ○上級テクニック
- ○ビットずれの応用
- ○回転変動の応用
- ○リセットの応用
- ○ハングアップの応用

応用編では知識編でみてきたそれぞれの知識が、どのようにプロテクトに生かされているかを例をあげて示します。また、プロテクトのテクニックを初歩のものから高度なものまで順に網羅し、各テクニックについて、原理、作成方法、チェック方法、そしてコピー方法などを解説します。

知識編第４章で説明したようにプロテクトには、作成手順を変更して通常でないフォーマットにするものと、誤差を利用するものがありますが、第１章 **基本テクニック** では前者の作成手順を変更するもののうち、変更するパラメータの少ないもの、比較的単純で他と複合して用いるための基本となるものを集めました。第２章 **上級テクニック** も同じく作成手順を変更するものですが、手順が複雑なもの、手順の推測がしにくいものなどを集めてあります。第３章以降は、知識編で取り上げてきたトピックの応用となっています。誤差を利用したもの、ＦＤＣの特殊な性質を利用したものなどを集めました。

作成方法 とは、前者の作成手順が問題となるプロテクトでの作成手順を示したものです。一般に、このような特殊な作成手順を施すのはディスケット中のある１つのトラックです。一部のプロテクトを除いて、そのような特殊処理されたトラックは、通常に使用できなくなりますから、知識編４．６に示したようにＦＡＴを変更するなどしてＯＳの管理下からはずした方が安全です。

あるトラックに特殊処理を施しただけでは、まだ完全なプロテクトになっていません。アプリケーションの実行中にそのトラックをチェックし、しかるべき作成手順に従って作られたものかどうかを調べ、そうでなければアプリケーションの実行を中断するようにプログラムを組んでおかねばなりません。**チェック方法** とは、そのような特殊な作成手順を経て作られたディスケットかどうかを調べる方法を示したものです。

最後に **コピー方法** を示しました。コピーといっても結局は同じものを作成することですから、そのプロテクトであることさえわかれば、後は作成方法どおりにすればいいわけで、問題は未知のトラックにどういうプロテクトがかかっているかを知ることです。そこでコピー方法では、どうやったら、そのプロテクトがかかっていることがわかるか、ということに重点を置きました。

作成方法とチェック方法、およびコピー方法では、必要に応じて実際のプログラム例を掲載しています。読者はこれらのプログラムの一部を変更、組み合わせることによって独自のプロテクトを作れることと思います。プログラムはＰＣ−９８００シリーズのＢＩＯＳ（知識編第３章）、および一部で拡張ＢＩＯＳ（付録１）を使っています。アセンブラに慣れていない読者は少し読みにくいかも知れませんが、日本語で詳しい注釈をつけていますので、これらのプログラムを通してだんだんとアセンブラもわかるようになるでしょう。

実際に、市販のソフトウェアにかけられているプロテクトの多くはいくつかのテクニックを複合して、コピーを困難にしています。一つひとつのテクニックはそれほど複雑でないものもありますが、それらを組み合わせることによって強力なプロテクトにすることができるわけです。読者の皆さんもここに紹介した断片的な各テクニックを組み合わせて、より強いプロテクトを完成されることを希望します。

プログラム例は、５インチ２ＤＤを例にとって作成してありますが、ここまで読んでこられた皆さんにとって、２ＨＤ用に変更することはわけのないことと思います。心配な方は必要に応じて知識編の第３章を御参照ください。

プログラム例の見方

プログラムはN₈₈−BASIC上で動作させることを前提として作成してあります。ただし、BIOSを使う必要がありますから、BASICだけでは記述できず機械語を併用することになります。そこで、各プログラム例は機械語のプログラムとそれを呼び出すBASICプログラムからなっています。機械語のプログラムは、開発のしやすさから **CP/M−86のアセンブラ** を用いて記述してあります。BASICのプログラムが伴っていないものは、機械語だけで動作するものです。

プログラムをそのまま使用する場合は、次のようにすると便利です。

① CLEAR文、DEF SEG文で機械語領域を定義する。

例 CLEAR ,&h1E00

DEF SEG=&h1E00

② BASICのモニタのEコマンドで機械語部分を入力する。

③ BASICのプログラム（ある場合）を入力する。

④ RUN

機械語だけの場合は、A=0：CALL Aとする。

実行する前にBASIC、機械語それぞれセーブしておくことをお勧めします。

機械語のプログラムに引数を渡すにはCALL文で行なえますが、ここでは、機械語プログラムを簡単にするため、BASICからPOKE文で直接、数値を書き換えています。また、機械語から値を受け取るにも関数PEEKを使います。

<モニタのAコマンドで機械語を入力する場合>

モニタのアセンブラとCP/M−86のアセンブラでは、一部表記が異なるので注意してください。以下に主な変更点を挙げます。

① ラベルが使えないので、アドレスでかくこと。

例 MOV CL,TRACK

→ MOV CL,BYTE [0100]; TRACKのアドレスを100Hとする

MOV BP,OFFSET DATA

→ MOV BP,0014 ; DATAのアドレスを14Hとする

OFFSETのついている場合はそのままアドレスを、ついていない場合は [ ] でくくって表わす。

② 10進数は16進に直すこと。

例 MOV BX,20 → MOV BX,0014

Hがついているものは16進ですから、Hをとるだけでよい。

③ セグメントオーバーライドは分離して1行前にかくこと。

例 MOV CS:SECTL,CH

→ CS: MOV [0100],CH; **SECTL**のアドレスを100Hとする

<MS−DOSのマクロアセンブラで機械語を入力する場合>

ほとんどそのまま使えますが変更すべき点は以下のとおりです。

① **SEGMENT**と**ENDS**で上下を囲むこと。**ASSUME**を入れること。

例
```
    CODE    SEGMENT BYTE PUBLIC
            ASSUME CS:CODE
      (ここにリストのプログラムが入る)
    CODE    ENDS
```

② 領域定義の方法が異なる。

例 DATA RS 1 → DATA DB ?

DATA RS 100H→ DATA DB 100H DUP (?)

1.1 基本テクニック 標準以外のセクタ長を使う

◎ 原 理

フォーマットするときのパラメータN（セクタ長）を変えるプロテクトです。N_{88}－ＢＡＳＩＣで使用するフロッピディスクのセクタ長はすべて256バイトときまっています。したがって、ユーティリティプログラム"backup.n88"では、すべて256バイトのセクタであることを前提としてコピーしようとしますから、もし256バイト以外のセクタにぶつかればDISK I/O ERRORとなって異常終了し、コピーはできません。

このことはN_{88}－ＢＡＳＩＣに限らず、各種ＯＳについてもあてはまり、そのＯＳの標準以外のセクタ長のものは、ＯＳ付属のバックアップユーティリティではコピーできません。

表１．１　各ＯＳの標準セクタ長

媒　体	N88-BASIC	MS-DOS	CP/M-86
２DD	256	512	256
２HDor８インチ	256	1024	256

◎ 作成方法

ＢＩＯＳ　フォーマットを使い、ディスケットのプロテクトすべき領域を256バイト以外のセクタ長でフォーマットします。ＢＩＯＳ　フォーマットでは倍密

度の場合、256バイト、512バイト、1024バイトのいずれかでフォーマットすることができますから、N₈₈－BASIC用なら512バイトか1024バイトでフォーマットすればよいでしょう。

プログラム例1

5インチ2DD、トラック1、サーフェス1を倍密度1024バイトでフォーマットします。

機械語

```
0000 B8715D          MOV AX,5D71H       ; 5DHはフォーマットコマンド
                                        ; 71Hは2DDのドライブ2
0003 BB1400          MOV BX,20          ; DATAの長さ
                                        ; 4バイト×5セクタ
0006 B90103          MOV CX,0301H       ; 03はセクタ長（1024バイト）
                                        ; 01はトラック=1
0009 BA0001          MOV DX,0100H       ; サーフェス=1
                                        ; 初期データ=00
000C 0E              PUSH CS
000D 07              POP ES
000E BD1400          MOV BP,OFFSET DATA
0011 CD1B            INT 1BH            ; BIOSコール
0013 CF              IRET
0014 01000103 DATA   DB 1,0,1,3         ; IDに書き込むデータ
0018 01000203        DB 1,0,2,3         ; 5セクタ分
001C 01000303        DB 1,0,3,3         ; 左からトラック番号、サーフェス
0020 01000403        DB 1,0,4,3         ; 番号、セクタ番号、セクタ長の順
0024 01000503        DB 1,0,5,3         ;
                 :
                     END
```

プロテクトをより有効とするには、フォーマットしてできたセクタにデータ、プログラム等を書き込んで使用するとよいでしょう。ＢＡＳＩＣコマンドでは無理ですからＢＩＯＳ　ライトデータを使います。

プログラム例２

前例でフォーマットしたトラックのセクタ３にデータを書き込みます。書き込むデータはアドレス１００Ｈ以降に格納されているものとします。

機械語

```
0000 B87155        MOV AX,5571H    ；５５Ｈはライトデータコマンド
0003 BB0004        MOV BX,400H     ；データの長さ 1024 バイト
0006 B90103        MOV CX,0301H    ；０３はセクタ長（1024 バイト）
                                   ；０１はトラック＝１
0009 BA0301        MOV DX,0103H    ；サーフェス＝１
                                   ；セクタ番号＝３
000C 0E            PUSH CS
000D 07            POP ES
000E BD0001        MOV BP,OFFSET DATA
                                   ；データ格納域アドレス
0011 CD1B          INT 1BH         ；ＢＩＯＳコール
0013 CF            IRET
                   ORG 100H
0100        DATA   RS 400H         ；データ格納域
                   END
```

◎ チェック方法

実際にデータを読み取る方法（Ａ）とＩＤのみ読む方法（Ｂ）とがあります。
（Ａ）　プロテクトを施したトラックのセクタを、書き込んだセクタ長を指定

してＢＩＯＳでリードし、エラーとならないこと（リターンコード＝０）を確認します。ここでリードしたデータをプログラムで有効なデータとして使用すればプロテクトはさらに確実になります。

（Ｂ） ＢＩＯＳ　リードＩＤで該当するトラックを読み、セクタ長コードが正しいかを確認します。

プログラム例３

Ａの方法。前例で書き込んだセクタをリードし、期待どおりのフォーマットか調べます。異常が発見された場合はこのサブルーチンから戻りません。

機械語

```
0000 B87156           MOV AX,5671H       ；５６Ｈはリード
0003 BB0004           MOV BX,400H        ；データの長さ 1024 バイト
0006 B90103           MOV CX,0301H       ；０３はセクタ長（1024 バイト）
                                         ；０１はトラック＝１
0009 BA0301           MOV DX,0103H       ；サーフェス＝１
                                         ；セクタ番号＝３
000C 0E               PUSH CS
000D 07               POP ES
000E BD0001           MOV BP,OFFSET      ；データ格納域アドレス
                      DATA
0011 CD1B             INT 1BH            ；ＢＩＯＳコール
0013 7201             JB ERROR           ；異常の場合ＥＲＲＯＲへ
0015 CF               IRET
              ;
0016 EBFE     ERROR:JMPSERROR            ；ここで無限ループ
              ;
                      ORG 100H
0100          DATA    RS 400H            ；データ格納域
                      END
```

◎ コピー方法

このフォーマットをコピーするためには、まず、そのトラックのセクタ長を知る必要があります。ＢＩＯＳ　リードＩＤでわかります。

1. ＢＩＯＳ　リードＩＤでセクタ長を求める。
2. 求めたセクタ長に従って　ＢＩＯＳでフォーマットをする。
3. 求めたセクタ長に従って　ＢＩＯＳでデータを読み込み、書き込む。

プログラム例4

トラック1のセクタ長を調べます。セクタ長は0から3のコードで表わされ、レジスタＣＨにはいりリターンします。

機械語

```
0000 B8715A            MOV AX,5A71H      ；５ＡＨはリードＩＤ
0003 B101              MOV CL,1          ；トラック＝1
0005 B600              MOV DH,0          ；サーフェス＝0
0007 CD1B              INT 1BH
0009 2E882E0001        MOV CS:SECTL,CH   ；読んだＮ（セクタ長）を
                                         ；ＳＥＣＴＬ（メモリ）に格納
000E CF                IRET
                 ;
                       ORG 100H
0100            SECTL  DB 0              ；ＣＨをここに格納しておき
                       END               ；ＢＡＳＩＣから読む
```

ＢＡＳＩＣ

```
10 A=0 : CALL A
20 PRINT"セクタ長コード=";PEEK(&h100)
```

1.2 基本テクニック 単密度を使う

◎ 原理

記録密度を変更するプロテクトです。フロッピディスクには単密度と倍密度の2種類の記録密度があり、倍密度のほうが2倍のデータを同じ面積に記録できます。現在では、倍密度を使うのが普通で単密度を使うシステムはほとんどありません。

領域は、無駄ですがプロテクトのために単密度を使うことができます。"backup.n 88"では倍密度を前提としており、単密度をサポートしていないため、単密度で書かれたトラックがあると読むことができず、Disk I/O Error となります。

◎ 作成方法

ＢＩＯＳのフィジカル・フォーマットでフォーマットするときに単密度を指定すれば、そのトラックのみ単密度でフォーマットすることができます。

プログラム例5

トラック5、サーフェス0を単密度 256 バイトでフォーマットします。

機械語

```
0000 B8711D  START: MOV AX,1D71H     ; 1DHは単密度フォーマット
```

```
0003 BB2400          MOV BX,36          ; 4バイト×9セクタ
0006 B90501          MOV CX,0105H       ; 01Hはセクタ長（256バイト）
                                        ; 05Hはトラック番号
0009 BA0000          MOV DX,0000H       ; サーフェス0
                                        ; 初期データ＝00
000C 0E              PUSH CS
000D 07              POP ES
000E BD1400          MOV BP,OFFSET DATA
                                        ; IDデータ格納域アドレス
0011 CD1B            INT 1BH            ; BIOSコール
0013 CF              IRET
               :
0014 05000101 DATA DB 5,0,1,1           ; IDに書き込むデータ
0018 05000201      DB 5,0,2,1           ; 9セクタ分
001C 05000301      DB 5,0,3,1           ; 左からトラック番号、サーフェス
0020 05000401      DB 5,0,4,1           ; 番号、セクタ番号、セクタ長の順
0024 05000501      DB 5,0,5,1           ;
0028 05000601      DB 5,0,6,1           ;
002C 05000701      DB 5,0,7,1           ;
0030 05000801      DB 5,0,8,1           ;
0034 05000901      DB 5,0,9,1           ;
               :
                   END
```

◎ チェック方法

単密度を指定して正常にリードできればOKです。

◎ コピー方法

コピーしたいトラックが単密度か倍密度かさえ分かれば、ＢＩＯＳのコマンドで倍密度の代わりに単密度を指定してやるだけです。問題は単密度か倍密度かの判定法です。一般に次のようなことがいえます。

1. 単密度を指定してリードＩＤを行なう。正常にリードできれば単密度である。
2. １でエラーの場合は倍密度を指定してリードＩＤを行なう。正常にリードできれば倍密度である。
3. ２でエラーの場合はアンフォーマット（後述）である。

プログラム例６

トラック５、サーフェス１が単密度か倍密度かアンフォーマットかを調べます。結果はＲＥＳＵＬＴ（１００Ｈ番地）にそれぞれ０、１、２で格納します。ＢＡＳＩＣでは、その値を基に結果を表示します。

機械語

```
0000 B8711A START: MOV AX,1A71H    ；単密度でリードＩＤ
0003 B105          MOV CL,5        ；トラック＝５
0005 B601          MOV DH,1        ；サーフェス＝１
0007 51            PUSH CX         ；ＣＸレジスタの保存
0008 52            PUSH DX         ；ＤＸレジスタの保存
0009 CD1B          INT 1BH         ；ＢＩＯＳコール
000B 5A            POP DX          ；ＤＸレジスタの復帰
000C 59            POP CX          ；ＣＸレジスタの復帰
000D 2EC60600010   MOV CS:RESULT,0 ；単密度のとき０
0013 7313          JNB RETN        ；ノーエラーならリターン
0015 B8715A        MOV AX,5A71H    ；倍密度でリードＩＤ
0018 CD1B          INT 1BH         ；ＢＩＯＳコール
```

```
001A 2EC60600010    MOV CS:RESULT,1 ; 倍密度のとき 1
0020 7306           JNB RETN        ; ノーエラーならリターン
0022 2EC60600010    MOV CS:RESULT,2 ; それ以外のとき 2
0028 CF   RETN:     IRET
          ;
                    ORG 100H
0100       RESULT   RS 1            ; 判定結果を格納する
                    END
```

BASIC

```
10 A=0 : CALL A
20 RESULT=PEEK(&h100)
30 IF RESULT=0 THEN PRINT "単密度です。" : END
40 IF RESULT=1 THEN PRINT "倍密度です。" : END
50 PRINT"アンフォーマットです。"
```



1.3 基本テクニック ディリーテッドレコードを使う

◎ 原　理

ディリーテッド・レコードといっても消されたレコードのことではありません。フォーマットは通常どおりでデータも存在します。データの書き込み時にデータ部の先頭にいっしょに書き込まれるアドレス・マークと呼ぶ１バイトのコードの値が異なるだけです。通常のセクタでは０ＦＢＨ、ディリーテッド・レコードでは０Ｆ８Ｈとなります。そこで、特定のセクタだけをディリーテッド・レコードにしておき、それをチェックすればプロテクトに利用できます。

◎ 作成方法

ディリーテッド・レコードを作るには、ＢＩＯＳの通常の書き込みコマンドを使う代わりに、ライトディリーテッド・データコマンドを使います。ディリーテッド・レコードはフォーマットを変えるものではないため、１セクタごとにディリーテッド・レコードにしたりしなかったりすることができます。

プログラム例７

トラック８、サーフェス０、セクタ７をディリーテッド・レコードとしてデータを書き込みます。１セクタ 256 バイト、倍密度とします。書き込むデータはアドレス１００Ｈ以降に格納されているとします。

機械語

```
0000 B87159 START: MOV AX,5971H        ；59Hはライトディリーテッド
                                       ；コマンド
0003 BB0001        MOV BX,100H         ；データの長さ、256バイト
0006 B90801        MOV CX,0108H        ；セクタ長256バイト
                                       ；トラック＝8
0009 BA0700        MOV DX,0007H        ；サーフェス＝0、セクタ＝7
000C 0E            PUSH CS
000D 07            POP ES
000E BD0001        MOV BP,OFFSET DATA
                                       ；データ格納域アドレス
0011 CD1B          INT 1BH             ；BIOSコール
0013 CF            IRET

                   ORG 100H
0100         DATA  RS 100H
                   END
```

◎ チェック方法

特定のセクタがディリーテッド・レコードかどうかをチェックします。BIOSで普通のリードを行ないリターンコードが10Hなら、ディリーテッド・レコードです。

プログラム例8

上例のセクタがディリーテッド・レコードかどうかを調べ、そうでないならばそこでストップし、ディリーテッド・レコードならリターンします。読み出したデータはアドレス100H以降にストアします。

機械語

```
0000 B87156 START: MOV AX,5671H      ;リードデータコマンド
0003 BB0001        MOV BX,100H       ;データの長さ
0006 B90801        MOV CX,0108H      ;データ長256バイト、トラック8
0009 BA0700        MOV DX,0007H      ;サーフェス0、セクタ7
000C 0E            PUSH CS
000D 07            POP ES
000E BD0001        MOV BP,OFFSET DATA
                                     ;データ格納域アドレス
0011 CD1B          INT 1BH           ;BIOSコール
0013 80FC10        CMP AH,10H        ;リターンコードが10Hか?
0016 75FE   LOOP:  JNE LOOP          ;ノーなら無限ループ
0018 CF            IRET
              :
                   ORG 100H
0100        DATA   RS 100H           ;データ格納域
                   END
```

◎コピー方法

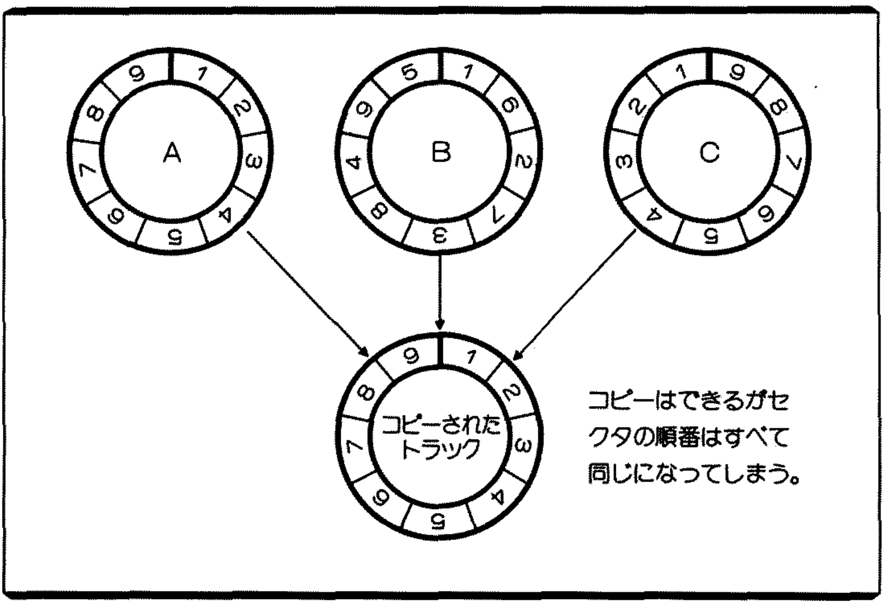
チェック方法と同じようにディリーテッド・レコードかどうかを調べ、ディリーテッド・レコードの場合(リターンコード＝１０H)はライトディリーテッド・データコマンド、普通のレコードの場合(リターンコード＝0)はライトデータコマンドを使ってデータを書き込んでやります。リードコマンドではリターンコードが１０Hになってもデータは読み出されています。

1.4 基本テクニック セクタの順番をチェック

◎ 原　理

1トラック中のセクタ番号はインデックスホールからの絶対的な位置ではなく、各セクタのＩＤ部に書かれたセクタ番号によって決まります。ですから、各セクタは必ずしもセクタ番号の順番に並んでいる必要はなく、各番号が１回ずつ現われる限りどんな並び方をしていても使用可能です。

図1．1　セクタの順番を変える



ですから、このフォーマットではデータはエラーなくコピーされます。ところがコピーされたディスケットではセクタの順番が決まった１つのパターンになってしまいます。そこで、普通は問題とならないこの違いをチェックしてプロテクトに利用します。

◎ 作成方法

通常のフォーマットをするだけですが、セクタの順番は通常使われない順番にしておきます。ＩＤに書き込むデータのうちＲの部分を変更することになります。

プログラム例９

上図Ｃの順番となるように、トラック５、サーフェス１をフォーマットします。ＭＳ－ＤＯＳを想定して１セクタ512バイト、９セクタ／トラックとします。

機械語

```
0000 B8715D   START:MOV AX,5D71H       ；倍密度フォーマット
0003 BB2400         MOV BX,36          ；４バイト×９セクタ
0006 B90502         MOV CX,0205H       ；セクタ長512バイト、トラック５
0009 BA0001         MOV DX,0100H       ；サーフェス１
                                       ；初期データ＝００
000C 0E             PUSH CS
000D 07             POP ES
000E BD1400         MOV BP,OFFSET DATA
                                       ；ＩＤデータの格納域アドレス
0011 CD1B           INT 1BH            ；ＢＩＯＳコール
0013 CF             IRET
                 ：
0014 05010902 DATA DB 5,1,9,2          ；セクタ番号が９，８，７，，の
0018 05010802      DB 5,1,8,2          ；順に並んでいる
001C 05010702      DB 5,1,7,2
```

```
0020 05010602          DB 5,1,6,2
0024 05010502          DB 5,1,5,2
0028 05010402          DB 5,1,4,2
002C 05010302          DB 5,1,3,2
0030 05010202          DB 5,1,2,2
0034 05010102          DB 5,1,1,2
                       END
```

◎ チェック方法

特定のセクタからトラックに書かれている順番に従ってセクタ番号を読み出し、フォーマット時と同じかどうかを比較します。書かれている順番に従ってセクタ番号を読み出すにはリードＩＤを使います。リードＩＤは現在ヘッドのある位置からサーチを始め、最初に見つけたＩＤを読むもので、続けて何度もリードＩＤをすることによって、書かれている順番にＩＤが読み出せるわけです。

プログラム例１０

上例で第９セクタの次はセクタ番号が８であることを確認します。ＯＫならリターン、違う場合はストップします。まずリードで第９セクタを読み出し、続いてすぐにリードＩＤを行なうことによって、すぐ次に続くセクタのＩＤを読み出すことができます。

機械語

```
0000 B87156 READ9: MOV AX,5671H       ;リードデータコマンド
0003 BB0002        MOV BX,200H        ;データ長512バイト
0006 B90502        MOV CX,0205H       ;セクタ長512バイト、トラック5
0009 BA0901        MOV DX,0109H       ;サーフェス1、セクタ9
000C 0E            PUSH CS
000D 07            POP ES
000E BD1E00        MOV BP,OFFSET DATA  ;データ格納域
```

```
0011 CD1B            INT 1BH
0013 B8715A READID:MOV AX,5A71H      ;リードID
0016 CD1B            INT 1BH
0018 80FA08          CMP DL,8        ;セクタ番号は8か?
001B 75FE   LOOP:    JNE LOOP        ;ノーなら無限ループ
001D CF              IRET
001E        DATA     RS 200H
                     END
```

◎ コピー方法

リードIDコマンドを続けて1トラックのセクタ数分だけ行なうことによって、1周全体のセクタ番号の順番がわかります。そして、その結果を用いてフォーマットします。

プログラム例11

指定したトラックのIDを指定した個数読み出し、表示します。記録密度は倍密度とします。

機械語

```
0000 0E              PUSH CS         ;DS←CS
0001 1F              POP DS          ;
0002 2E8A1E0201      MOV BL,KOSU
0007 BE0301          MOV SI,OFFSET   IDAREA
            LOOP:                    ;読んだIDの格納域アドレス
000A 2E8A0E0001      MOV CL,TRACK
000F 2E8A360101      MOV DH,SURFACE
0014 B8715A          MOV AX,5A71H    ;リードID
0017 CD1B            INT 1BH         ;BIOSコール
0019 880C            MOV [SI],CL     ;IDのCを格納
```

```
001B 46              INC SI
001C 8834            MOV [SI],DH      ；ＩＤのHを格納
001E 46              INC SI
001F 8814            MOV [SI],DL      ；ＩＤのRを格納
0021 46              INC SI
0022 882C            MOV [SI],CH      ；ＩＤのNを格納
0024 46              INC SI
0025 FECB            DEC BL           ；個数デクリメント
0027 75E1            JNE LOOP         ；0でないならLOOPへ
0029 CF              IRET
                 ：
                     ORG 100H
0100    TRACK        RS 1             ；トラック番号
0101    SURFACE      RS 1             ；サーフェス番号
0102    KOSU         RS 1             ；読むＩＤの個数
             ：
0103    IDAREA       RS 100           ；読んだＩＤの格納域
                     END
```

ＢＡＳＩＣ

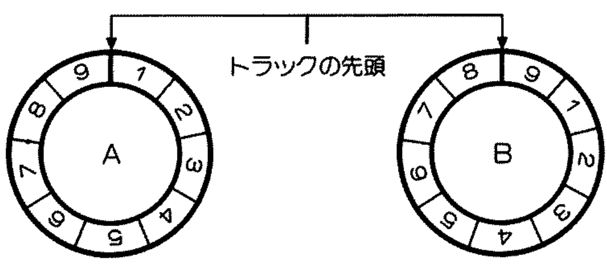
```
10 INPUT"トラック=";TR  : POKE &h100,TR  ；機械語のワークへセット
20 INPUT"サーフェス=";SUR: POKE &h101,SUR ；機械語のワークへセット
30INPUT" ＩＤの個数=";KO: POKE &h102,KO  ；機械語のワークへセット
40 A=0 : CALL A
50 PRINT "C  H  R  N"
60 FOR I=0 TO KO-1
70 FOR J=0 TO 3
80 PRINT PEEK(I * 4+J+&h103);
90 NEXT : PRINT
100 NEXT
```

1.5 基本テクニック セクタの位置をチェック

◎原　理

前節のテクニックはセクタの書かれている順番をチェックするもので、絶対的な位置までは調べませんでした。しかし、それでは次図のA，Bのようにフォーマットされたディスケットを別のものとして区別することはできません。なぜなら、両方とも隣り合うセクタの関係がまったく同じだからです。

図1．2　順番は同じでも位置が違う



リードＩＤを実行するときにはインデックス・ホールの位置を基準にするということはされませんから、1．4節でやったように、もとのディスケットから読み取ったＩＤの順序に従ってフォーマットしても、この図のような違いのあるコピーディスケットができることもあるわけです。

そこで、インデックス・ホールの位置を基準とした各セクタの絶対的な位置を

チェックすることによってA，Bを違うものとして見分ける方法が考えられます。

◎―――作成方法

前節1．4とまったく同じ方法で作成すればよいでしょう。普通に見せかけるため図1．2（B）のようにするとよいでしょう。

◎―――チェック方法

この方法では、各セクタの位置をチェックするためにトラックの先頭を基準とするので、まずトラックの先頭、つまり、インデックス・ホールの位置を見つけなくてはなりません。ＢＩＯＳコマンドには「インデックス・ホールの位置を見つける」というものはありませんが、次の方法でインデックス・ホールの位置をみつけることができます。

1．　記録密度が倍密度のとき、単密度でリードまたはリードＩＤを行なう。
2．　記録密度が単密度のとき、倍密度でリードまたはリードＩＤを行なう。

これらの命令終了直後には、ヘッドはトラックの先頭に位置付けられています。なぜなら、記録密度が実際と異なるように指定しているために、ＦＤＣはこの命令を実行してもまったくＩＤを見つけることができず、２回インデックス・ホールを検出したところで異常終了となります。つまり、わざと読めない記録密度で読ませるように指定して異常終了させ、その終了のタイミング（インデックス・ホール検出直後）を利用するのです。

このようにしてインデックス・ホールの位置を見つけることができるので、上記命令を実行し、その直後に正常な記録密度でリードＩＤを行なえば、インデックス・ホール直後のセクタのＩＤを読めるわけです。上図Ａでは１、Ｂでは９となります。

プログラム例１２

トラック５、サーフェス０の先頭のセクタ番号を求め表示します。機械語のプログラムではメモリ（１００番地）にいったん格納し、ＢＡＳＩＣで読み出して表示します。記録密度は倍密度とします。

機械語

```
0000 B8711A  RIDSNG:MOV AX,1A71H     ；単密度リードＩＤ（頭出し）
0003 B105           MOV CL,5         ；トラック５
0005 B600           MOV DH,0         ；サーフェス０
0007 CD1B           INT 1BH
0009 B8715A  RIDDBL:MOV AX,5A71H     ；倍密度リードＩＤ
000C CD1B           INT 1BH
000E 2E88160001     MOV CS:RESULT,DL ；セクタ番号をＲＥＳＵＬＴに格納
0013 CF             IRET
              :
                    ORG 100H
0100         RESULT DS 1
                    END
```

◎ コピー方法

チェック方法で述べた手法を用います。まず、トラックの頭出し（インデックス・ホールの位置の検出）をした後、リードＩＤをセクタの数だけ実行し、インデックス・ホールから始まる１周全体のセクタ番号を読み取り、それを用いて、フォーマットします。もちろん、これをトラック毎に行なう必要があります。

プログラム例13

指定したトラックのＩＤを指定した個数だけトラックの先頭から読み出し表示します。記録密度は倍密度とします。

機械語

```
0000 0E              PUSH CS           ；DS←CS
0001 1F              POP DS            ；
0002 2E8A0E0001      MOV CL,TRACK
0007 2E8A360101      MOV DH,SURFACE
000C B8713A          MOV AX,3A71H      ；単密度リードＩＤ
                                       ；エラーリトライなし
000F CD1B            INT 1BH           ；ＢＩＯSコール
0011 2E8A1E0201      MOV BL,KOSU
0016 BE0301          MOV SI,OFFSET IDAREA   ；読んだＩＤの
               LOOP:                        格納域アドレス
0019 2E8A360101      MOV DH,SURFACE
001E B8714A          MOV AX,4A71H      ；倍密度リードＩＤ，シークなし
0021 CD1B            INT 1BH
0023 880C            MOV[SI],CL        ；ＩＤのCを格納
0025 46              INC SI
0026 8834            MOV [SI],DH       ；ＩＤのHを格納
0028 46              INC SI
0029 8814            MOV [SI],DL       ；ＩＤのRを格納
002B 46              INC SI
002C 882C            MOV [SI],CH       ；ＩＤのNを格納
002E 46              INC SI
002F FECB            DEC BL            ；個数デクリメント
0031 75E6            JNE LOOP          ；0でないならLOOPへ
0033 CF              IRET
               ；
                     ORG 100H
```

```
0100    TRACK     RS 1          ;トラック番号
0101    SURFACE   RS 1          ;サーフェス番号
0102    KOSU      RS 1          ;読むＩＤの個数

0103    IDAREA    RS 100        ;読んだＩＤの格納域
                  END
```

ＢＡＳＩＣ

```
10 INPUT"トラック =";TR       : POKE &h100,TR  ;機械語のワークへセット
20 INPUT"サーフェス =";SUR    : POKE &h101,SUR ;機械語のワークへセット
30 INPUT"ＩＤの個数 =";KO     : POKE &h102,KO  ;機械語のワークへセット
40 A=0 : CALL A
50 PRINT "C  H  R  N"
60 FOR I=0 TO KO-1
70 FOR J=0 TO 3
80 PRINT PEEK (I * 4+J+&h103);
90 NEXT : PRINT
100 NEXT
```

1.6 基本テクニック IDのCが実際と異なる

◎ 原　理

各セクタのＩＤ部にはＣ（トラック番号）、Ｈ（サーフェス番号）、Ｒ（セクタ番号）、Ｎ（セクタ長コード）の４つの情報が書き込まれています。セクタ番号に関してはこのＲが唯一の情報ですが、トラック番号、サーフェス番号に関してはそれぞれ物理的に位置が決まっていますから、ＩＤに書かれている情報は確認の意味しかありません。そこで、このＣの値を実際のトラック番号と異なる値で書き込めば、ちょっとした工夫で読み書きは可能ですが、通常は、物理的なトラックの位置とＣの値は等しいことが前提となっていますから、単純にはコピーできません。

◎ 作成方法

フォーマット時に与えるＩＤデータに手を加えるだけです。ＩＤデータのＣの値に実際のトラック番号ではない数を入れてやればいいのです。１トラック全体にわたってＣの値を変えてもいいですし、１セクタだけＣの値を変えてもかまいません。

このようにフォーマットされたセクタにデータを書き込むには少し工夫がいります。ＢＩＯＳではシーク（ヘッドを目的のトラックの上にもっていく）と書き込みを１命令で行なうことができますが、その場合は物理的トラックの位置＝Ｃの値として扱われますから、ここではシークと書き込みは別々に行なう必要があ

ります。

① まずシークコマンドでヘッドを目的とするトラック上に移動します。このときは物理的な実際のトラック番号を指定します。

② 次にライトコマンドをシーク０ＦＦで実行します。このときはフォーマット時にＩＤデータとして書き込んだＣの値を指定します。

ＦＤＣはＩＤの４情報がすべて指定と一致するセクタを探しますから、このようにしてやらないといけないのです。

プログラム例１４

ＩＤのＣが物理トラック番号と異なるセクタに、データを書き込みます。物理トラック番号が５、ＩＤのＣが６の場合を例にとります。倍密度でサーフェスは０、セクタは３、セクタ長は 256バイト（Ｎ＝１）とします。また、書き込むデータは１００Ｈ番地以降に格納されているものとします。

機械語

```
0000 B87110    MOV AX,1071H   ;シークコマンド
0003 B105      MOV CL,5       ;トラック＝5
0005 CD1B      INT 1BH        ;ＢＩＯＳコール
0007 B87145    MOV AX,4571H   ;ライトデータ、シークなし
000A BB0001    MOV BX,100H    ;データ長 256バイト
000D B90601    MOV CX,0106H   ;セクタ長 256バイト
                              ;トラック（ＩＤのＣ）＝6
0010 BA0300    MOV DX,0003H   ;サーフェス＝0、セクタ＝3
0013 0E        PUSH CS        ;ES←CS
0014 07        POP ES         ;
0015 BD0001    MOV BP,100H    ;データの格納域アドレス
0018 CD1B      INT 1BH        ;ＢＩＯＳコール
001A CF        IRET
               END
```

◎ チェック方法

ＩＤだけを読んでチェックする方法と，実際にデータを読み込む方法とがあります。

1．　ＩＤだけを読む方法

トラック全体に渡ってＣの値を変えた場合、この方法で簡単にチェックが行なえます。リードＩＤを実行し読んだＣの値が期待どおりかどうか調べればいいのです。リードＩＤではＩＤの内容はまったくチェックしませんから、物理的に指定したトラック中で最初に出会ったセクタのＩＤを返してくれるのです。

2．　データを読み込む方法

データを読み込むには作成時と同じ手順を踏む必要があります。まず、シークを行ないヘッドを移動させてから、シーク０ＦＦでリードコマンドを与えます。正常にリードでき、作成時にそのセクタに書き込んだはずのデータと一致すればＯＫです。

プログラム例１５

ＩＤのＣが物理トラック番号と異なる場合のデータの読み込み方法を示します。物理トラック番号が５、ＩＤのＣが６の場合を例にとります。倍密度でサーフェスは０、セクタは３、セクタ長は 256バイト（Ｎ＝１）とします。また読み込んだデータは、１００Ｈ番地以降に格納するものとします。

機械語

```
0000 B87110        MOV AX,1071H     ；シークコマンド
0003 B105          MOV CL,5         ；トラック＝5
0005 CD1B          INT 1BH          ；ＢＩＯＳコール
0007 B87145        MOV AX,4671H     ；リードデータ、シークなし
000A BB0001        MOV BX,100H      ；データ長 256バイト
000D B90601        MOV CX,0106H     ；セクタ長 256バイト
                                    ；トラック（ＩＤのＣ）＝6
```

```
0010 BA0300      MOV DX,0003H    ；サーフェス＝0
                                 ；セクタ＝3
0013 0E          PUSH CS         ；ES←CS
0014 07          POP ES          ；
0015 BD0001      MOV BP,100H     ；データの格納域アドレス
0018 CD1B        INT 1BH         ；BIOSコール
001A CF          IRET
                 END
```

◎ コピー方法

Cの値さえ求まれば後は上と同じです。

1. リードIDでCの値を読む。
2. その結果に基づいてフォーマット。
3. チェック方法の2に従いデータをリード。
4. 作成方法に従いデータをライト。

1.7 基本テクニック ＩＤのＨが実際と異なる

◎ 原理

ＩＤに書き込むＨに手を加えるプロテクトです。ヘッド番号は通常フロッピディスクの表が０、裏が１と決まっていますが、各セクタのＩＤ情報中に書き込むヘッド番号（Ｈ）にはトラック番号（Ｃ）と同じく、実際の表裏とは関係なく、任意の値を付けることができます。普通は表が０、裏が１というのが前提ですから、Ｈを変えただけでもコピーできなくなるのです。

◎ 作成方法

フォーマットするだけならＩＤに書き込むデータを変えるだけですが、そのセクタにデータを書き込もうとすると問題があります。ＢＩＯＳでは、Ｃと同じく物理的なヘッド番号の指定とＩＤの中のＨの指定をＤＨレジスタ１つで兼用しているため、Ｈを変えると物理的ヘッドまで変わってしまうからです。といってもヘッドは０と１の２種類で、Ｈに偶数を指定したときは０が、奇数のときは１が使われる物理的ヘッド番号です。

表に偶数番号、裏に奇数番号を指定する場合には特に問題はありませんが、逆に表に奇数番号、裏に偶数番号を指定するとアクセスできないのが普通です。この場合は、レジスタＡＬのＨＲビット（知識編第３章ＢＩＯＳの知識参照）を１にして物理的ヘッド番号を反転させます。

プログラム例１６

ＩＤのＨが物理ヘッドと異なる場合の書き込み方法を示します。物理ヘッド（サーフェス）０でＩＤのＨが７の場合を例にとります。倍密度でトラック５、セクタは３、セクタ長は 256バイト（N＝１）とします。また書き込むデータは１００Ｈ番地以降に格納されているとします。ＩＤのＨと物理ヘッドの最下位ビットが異なるためALのHRビットを１にします。

機械語

```
0000 B87555      MOV AX,5575H    ;ライトデータ、HRビット＝1
0003 BB0001      MOV BX,100H     ;データ長256バイト
0006 B90501      MOV CX,0105H    ;セクタ長256バイト
                                 ;トラック＝5
0009 BA0307      MOV DX,0703H    ;ＩＤのH＝7、セクタ＝3
000C 0E          PUSH CS         ;ES←CS
000D 07          POP ES          ;
000E BD0001      MOV BP,100H     ;データの格納域
0011 CD1B        INT 1BH         ;BIOSコール
0013 CF          IRET
```

◎チェック方法

ここでも、１．６節同様ＩＤだけを読んでチェックする方法と、データを読み込む方法が考えられます。

１．　ＩＤだけを読む方法

トラック全体に渡ってＨの値を変えた場合、そのトラック中のどれかひとつのセクタについてＩＤのＨの値を調べれば十分です。つまり、リードＩＤを実行し、読んだＨの値が期待どおりかどうか調べればよいのです。

２．　データを読み込む方法

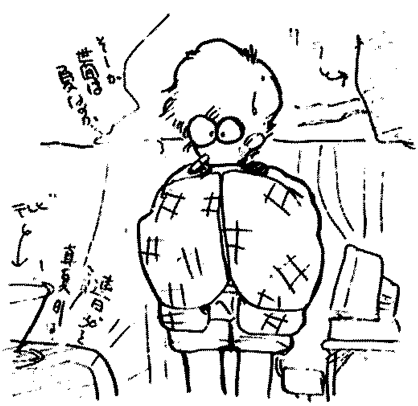
ＢＩＯＳのリードデータを使えばよいわけですが、Ｈが表で奇数か裏で偶数の場合は、レジスタＡＬのＨＲビットを１にしなければなりません。

正常にリードでき、作成時にそのセクタに書き込んだはずのデータと一致すればＯＫです。

◎――コピー方法

1．6節の方法とほぼ同様にしてコピーできます。

1．　リードＩＤでＨの値を読む。
2．　その結果に基づいてフォーマット。
3．　チエック方法の２に従いデータをリード。
4．　作成方法に従いデータをライト。

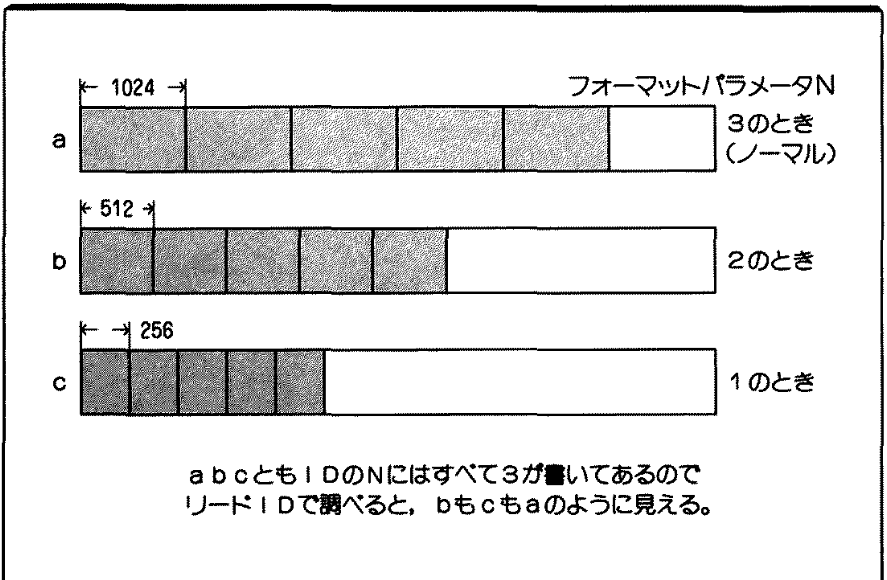


1.8 基本テクニック IDのNが実際と異なる

◎ 原　理

フォーマット時のセクタ長のパラメータNと、ＩＤに書き込むNを異なる値にするものです。１．１節に紹介したのもセクタ長を変えるものですが、そのときは、ＩＤのNもフォーマットパラメータのNも同時に同じ値に変えたものですが、ここでは、一方だけを変えます。

図１．３　ＩＤのＮが実際のセクタ長と違う場合



こうするとリードＩＤを行なっても本当のセクタ長を知ることができず、同じフォーマットが作れなくなります。Ｎとセクタ長の関係は知識編表２．５を参照してください。

◎作成方法

フォーマット時に与えるＩＤデータのＮの部分を変えるだけです。しかし、セクタ数を通常のままにしておくと、セクタ数からセクタ長の見当がつきますから、セクタ数も変えた方が効果的です。

プログラム例１７

図１．３（Ｃ）のフォーマットを行ないます。実際のセクタ長は256バイト（Ｎ＝１）ですが、ＩＤのＮはすべて３です。セクタ数は５とします。トラックは８、サーフェスは１とします。

機械語

```
0000 B8715D          MOV AX,5D71H     ；フォーマットコマンド
0003 BB1400          MOV BX,20        ；５セクタ×４バイト
0006 B90801          MOV CX,0108H     ；セクタ長256バイト
                                      ；トラック＝８
0009 BA0001          MOV DX,0100H     ；サーフェス＝０、初期データ＝00
000C 0E              PUSH CS          ；ES←CS
000D 07              POP ES           ；
000E BD1400          MOV BP,OFFSET  DATA
                                      ；ＩＤデータの格納域アドレス
0011 CD1B            INT 1BH          ；ＢＩＯＳコール
0013 CF              IRET
                 ：
0014 08010103 DATA DB 8,1,1,3         ；ＩＤに書き込むデータ
0018 08010203      DB 8,1,2,3         ；左からトラック番号（Ｃ）
```

```
0001C 08010303      DB 8,1,3,3      ；サーフェス番号（H）、
0020 08010403       DB 8,1,4,3      ；セクタ番号（R）、
0024 08010503       DB 8,1,5,3      ；セクタ長コード（N）の順
                    END             ；Nが3であることに注目
```

◎ チェック方法

このフォーマットはリードＩＤを行なってもセクタ長がわからないようにするフォーマットですから、単にリードＩＤを行なってもチェックはできません。

1．データを読んでCRCエラーを確認する方法

データをリードするときは、フォーマット時のNに関係なく、ＩＤに書かれているNに従って読み込むセクタ長が決まります。実際と異なるときはCRCが一致するはずはなく、CRCエラーが出ます。そこでCRCエラー（リターンコード＝B０H）が出ればNが実際と違うことがわかります。

しかし、この方法では違うことがわかるだけでフォーマット時のNがいくつだったかわかりません。

2．データのパターンから調べる方法

リードダイアグノスティックでトラックの先頭から１トラック全体を読みます。２DDならN＝６、２HDならN＝７とします。フォーマットしただけならデータは１セクタ内のすべてで同じ値となっているのが普通です。そこで読んだデータの先頭からどこまで同じデータが続くかを調べます。同じデータの続いた長さがフォーマット時のセクタ長です。

◎ コピー方法

チェック方法と同じようにしてフォーマット時のセクタ長を求めます。あとはリードＩＤで読んだＩＤをデータにしてフォーマットするだけです。

1.9 基本テクニック 標準より少ないセクタ数

◎ 原　理

1トラックに入るセクタの数の標準は、媒体、記録密度、セクタ長によって表1．2のように決まっています。このプログラム例はセクタ数を減らすプロテクトです。"backup.n88"などのバックアッププログラムは、必ず標準のセクタ数を全部コピーしようとしますから、セクタ数を減らすと途中でエラー終了してしまうのです。

表1．2　標準のセクタ数

記録密度	倍密度			単密度		
媒体＼セクタ長	256	512	1024	128	256	512
2DD	16	9	5	16	9	5
2HDor 8インチ	26	15	8	26	15	8

◎ 作成方法

これはフォーマット時のパラメータSC（セクタ数）を少なくすることに相当します。しかし、BIOSではSCを指定できません。その代わりデータ長を短くしてやることによって、FDCにTC入力が入り途中で終了するため同じ結果を得られます。フォーマット時に指定するデータ長はIDに書き込むデータの長さですが、1セクタあたり4バイトですから、nセクタにしたければデータ長を

n×4に設定すればいいのです。

プログラム例１８

標準より少ないセクタ数を作る例です。セクタ長 256 バイトの場合は標準は２DDで 16 ですが、それを 12 にします。トラックは 20（１４H）、サーフェスは０とします。

機械語

```
0000 B8715D           MOV AX,5D71H       ；フォーマットコマンド
0003 BB3000           MOV BX,48          ；12 セクタ×４バイト
0006 B91401           MOV CX,0114H       ；セクタ長256 バイト
                                         ；トラック=20
0009 BA0000           MOV DX,0000H       ；サーフェス＝０、初期データ=00
000C 0E               PUSH CS            ；ES←CS
000D 07               POP ES             ；
000E BD1400           MOV BP,OFFSET DATA；ＩDデータの格納域アドレス
0011 CD1B             INT 1BH            ；BIOSコール
0013 CF               IRET
0014 14000101 DATA    DB 20,0,1,1        ；ＩDに書き込むデータ
0018 14000202         DB 20,0,2,2
001C 14000302         DB 20,0,3,2
0020 14000402         DB 20,0,4,2
0024 14000502         DB 20,0,5,2
0028 14000602         DB 20,0,6,2
002C 14000702         DB 20,0,7,2
0030 14000802         DB 20,0,8,2
0034 14000902         DB 20,0,9,2
0038 14000A02         DB 20,0,10,2
003C 14000B02         DB 20,0,11,2
0040 14000C02         DB 20,0,12,2
```

◎ チェック方法

セクタ数を少なくしたのですから、標準なら存在すべきセクタがないことになります。そのセクタをリードしようとしてエラー（リターンコード＝C０H）となることを確認すればいいでしょう。

◎ コピー方法

リードＩＤをすることによってわかります。何回もリードＩＤをすればあるパターンで繰り返されますから、繰り返しの１回分が１周に相当し、セクタ数がわかります。

1.10 基本テクニック 標準より多いセクタ数

◎原理

前節とは反対に標準よりセクタ数をふやそうというものです。減らすのは簡単ですが、ふやすのは少しやっかいです。そもそもできるだけたくさんデータを記録したいのですから、標準は最大値になっていると考えられます。ただし、セクタとセクタの間のギャップを適度にとった場合の最大値で、もっとセクタ間を詰めて押し込めばこれ以上いれることも可能です。

1トラックの構成は知識編図2．1に示したとおりです。セクタとセクタの間にはGAP3という緩衝帯がありますが、これを短くすればもっとセクタ数をふやすことができます。入るセクタ数は次の計算で求められます。総バイト数は2DDで約6250バイト、2HDまたは8インチで10416バイトです。GAP3は1から256まで変えることができます。

倍密度の場合

(62＋セクタ長）×セクタ数＋GAP3×（セクタ数－1）＝総バイト数

式（1－1）の関係から

$$\text{セクタ数}=\frac{\text{総バイト数}-\text{GAP}3}{62+\text{セクタ長}+\text{GAP}3} \cdots\cdots \text{式（1－2）}$$

となります。

単密度の場合

（33＋セクタ長）×セクタ数＋ＧＡＰ３×（セクタ数－１）＝総バイト数

式（１－３）の関係から

$$\text{セクタ数} = \frac{\text{総バイト数} - \text{GAP3}}{33 + \text{セクタ長} + \text{GAP3}} \cdots\cdots\cdots\cdots\cdots\cdots \text{式（1－4）}$$

となります。

このようにして求められたセクタ数は１トラックに入れることのできる最大値ですから、その値を超えない範囲で標準よりも多くセクタ数を設定すればよいわけです。

◎ 作成方法

フォーマット時に指定するSC（セクタ数）とGAP３の値を上の方法で求めて、セクタ数が標準よりも多くなるようにフォーマットします。つまりSCは多くしてGAP３を小さくします。ＢＩＯＳではSC、GAP３の値を変えることができないのが普通なので、知識編３．３で説明したようにするか、付録の拡張ＢＩＯＳを使ってください。

プログラム例１９

標準より多いセクタ数を作る例です。セクタ長 256バイトの場合は標準は２DDで16ですが、それを17にします。そのためGAP３は短くして40（２８H）にします。拡張ＢＩＯＳを使って実現します。トラックは20（１４H）、サーフェスは０とします。

機械語

```
0000 B8715D        MOV AX,5D71H    ；フォーマットコマンド
0003 BB4400        MOV BX,68       ；17セクタ×４バイト
```

```
0006 B91401            MOV CX,0114H       ；セクタ長256バイト，トラック=20
0009 BA0000            MOV DX,0000H       ；サーフエス=0、初期データ=00
000C 0E                PUSH CS            ；ES←CS
000D 07                POP ES             ；
000E BD1A00            MOV BP,OFFSET DATA
                                          ；IDデータの格納域
0011 BE2811            MOV SI,1128H       ；GPL=28H(40),
                                          ；SC=11H(17)
0014 BFFFFF            MOV DI,0FFFFH      ；リセット使わない
0017 CD40              INT 40H            ；拡張BIOSコール
0019 CF                IRET
001A 14000101 DATA DB 20,0,1,1            ；IDに書き込むデータ
001E 14000201          DB 20,0,2,1
0022 14000301          DB 20,0,3,1
0026 14000401          DB 20,0,4,1
002A 14000501          DB 20,0,5,1
002E 14000601          DB 20,0,6,1
0032 14000701          DB 20,0,7,1
0036 14000801          DB 20,0,8,1
003A 14000901          DB 20,0,9,1
003E 14000A01          DB 20,0,10,1
0042 14000B01          DB 20,0,11,1
0046 14000C01          DB 20,0,12,1
004A 14000D01          DB 20,0,13,1
004E 14000E01          DB 20,0,14,1
0052 14000F01          DB 20,0,15,1
0056 14001001          DB 20,0,16,1
005A 14001101          DB 20,0,17,1
                       END
```

◎ チェック方法

作成時に設定したセクタ数と実際にディスケットに存在する、すなわち、読み出すことのできたセクタ数とが一致することを確認します。

もちろん、標準セクタ数を超えた部分にセクタが存在しなければ読み出しの際にエラーになりますから、特にチェックする必要もないともいえます。

◎ コピー方法

前節と同じようにリードＩＤを繰り返し使えばあるところで同じＩＤが読み出されますから、セクタ数を求めてください。あとは、作成方法と同じです。

1.11 基本テクニック アンフォーマットを使う

◎ 原理

通常と違うという点ではまったくフォーマットをしないというのもプロテクトになり得ます。つまり、1つもセクタが存在しないトラックを作ることをアンフォーマットといいます。

5インチのディスケットは、一般にフォーマッティングしていない状態で売られていますが、8インチのディスケットは普通IBMフォーマットでフォーマッティングされて売られています。したがって、5インチの場合はフォーマットしないトラックを残すだけでもよいのですが、8インチではフォーマットされたトラックを、もういちどアンフォーマットの状態にもどさなければなりません。

そこでここでは、正常なセクタがひとつもないようなフォーマットのかけかたを考えます。

◎ 作成方法

2DDではNを6以上に、2HDでは7以上にしてフォーマットします。つまり実際には、フォーマットはおこないますが、結果的にアンフォーマットとみなされる状態にすることができるのです。

2DDをN＝6（8192バイト）でフォーマットするとDATA部の途中でセクタの長さがトラック1周の長さ（6250バイト）を超えてしまい、2周目にはいって自分自身のID部をもつぶしてしまいます。つまり、まったくIDがなく

なるわけで、これはソフト的に見てアンフォーマットとまったく同じ状態です。

プログラム例２０

アンフォーマットを作ります。２ＤＤでは、Ｎ＝６にしてフォーマットすればアンフォーマットです。トラックは30（１ＥＨ）、サーフェスは１とします。

機械語

```
0000 B8715D        MOV AX,5D71H     ;フォーマットコマンド
0003 BB0400        MOV BX,4         ;セクタ数＝1
0006 B91E06        MOV CX,061EH     ;セクタ長8192バイト（N＝6）
                                    ;トラック=30
0009 BA0001        MOV DX,0100H     ;サーフェス＝1、初期データ=00
000C 0E            PUSH CS          ;ES←CS
000D 07            POP ES           ;
000E BD0001        MOV BP,100H
0011 CD1B          INT 1BH          ;BIOSコール
0013 CF            IRET
                   END
```

◎チェック方法

リードＩＤを単密度、倍密度の両方で実行し、ともにエラー（リターンコードＥ０Ｈ）となれば、そのトラックでセクタをひとつもみつけられない、つまりはアンフォーマットということがわかります。

◎コピー方法

チェック方法と同じようにしてアンフォーマットと判定されたら、作成方法に従います。

1.12 基本テクニック オーバートラックを使う

◎ 原　理

５インチ２ＤＤでは、片面当たり０から79の80トラックがありますが、機械的には、79トラックの内側にさらに数トラック分、ヘッドを移動させることができます。これを利用して通常では使われない80,81,82,83などのトラックを使用することをオーバートラックといいます。トラック79までしかコピーしないプログラムではこの部分がコピーされないわけで、プロテクトになるのです。２ＨＤでは75までが標準ですから、さらにかなりのトラックを使えます。８インチでは機械的にあまり内側へ動かないようになっていますので、注意してください。

◎ 作成方法

２ＤＤでは80以上のトラックに、２ＨＤでは76以上のトラックにフォーマットしてデータを書き込む。トラック番号を変えるだけであとは普通とまったく同じです。

◎ チェック方法

書き込んだデータを読んでみるとよいでしょう。

◎ コピー方法

常に機械的な限界までトラックをチェックすることが必要です。自分のドライブ装置が何トラックまで使えるかをチェックしておくとよいでしょう。

プログラム例21

あなたのドライブ装置が何トラックまで使えるかを調べるプログラムです。このプログラムでは、まずトラック100をフォーマットします。といっても実際は100トラックまでありませんから、結局、最大のトラックがフォーマットされることになります。次にトラック80から順にリードIDを行なって初めて見つけたフォーマットされたトラックが最大トラックとなります。限界以上に移動させるとトラック番号が狂ってしまいますので、リードIDの前にはリキャリブレイトを行なって正しいトラックに戻します。

機械語

```
0000 B8715D          MOV AX,5D71H    ;フォーマットコマンド
0003 BB0400          MOV BX,4        ;セクタ数=1
0006 B96401          MOV CX,0164H    ;セクタ長256バイト
                                     ;トラック=100
0009 BA0000          MOV DX,0000H    ;サーフェス=0、初期データ=00
000C 0E              PUSH CS         ;ES←CS
000D 07              POP ES          ;
000E BD0001          MOV BP,100H
0011 CD1B            INT 1BH         ;BIOSコール
0013 B87107          MOV AX,0771H    ;リキャリブレイトコマンド
0016 CD1B            INT 1BH         ;BIOSコール
0018 B150            MOV CL,80       ;トラック=80
001A B600      LOOP: MOV DH,0        ;サーフェス=0
001C B8717A          MOV AX,7A71H    ;リードID, エラーリトライなし
001F 51              PUSH CX         ;CXレジスタ保存
0020 CD1B            INT 1BH         ;BIOSコール
```

```
0022 59                  POP CX           ；CXレジスタ復帰
0023 FEC1                INC CL           ；トラック=トラック+1
0025 72F3                JB LOOP          ；セクタが見つからなかったとき
                                          ；LOOPへ
0027 FEC9                DEC CL           ；トラック=トラック−1
0029 2E880E0001          MOV CS:TRACK,CL  ；最大トラック番号を
                                            TRACKへ格納
002E CF                  IRET
                                          ；
                         ORG 100H
0100        TRACK        RS 1             ；求めた最大トラック番号を格納
                         END
```

BASIC

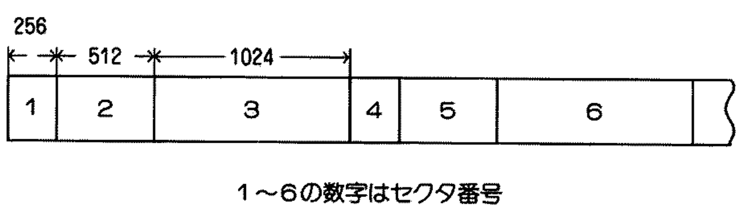
```
10 A=0 : CALL A
20 PRINT"最大トラック番号=";PEEK(&h100)
```

2.1 上級テクニック 1トラック中に異なるセクタ長が混在

◎ 原　理

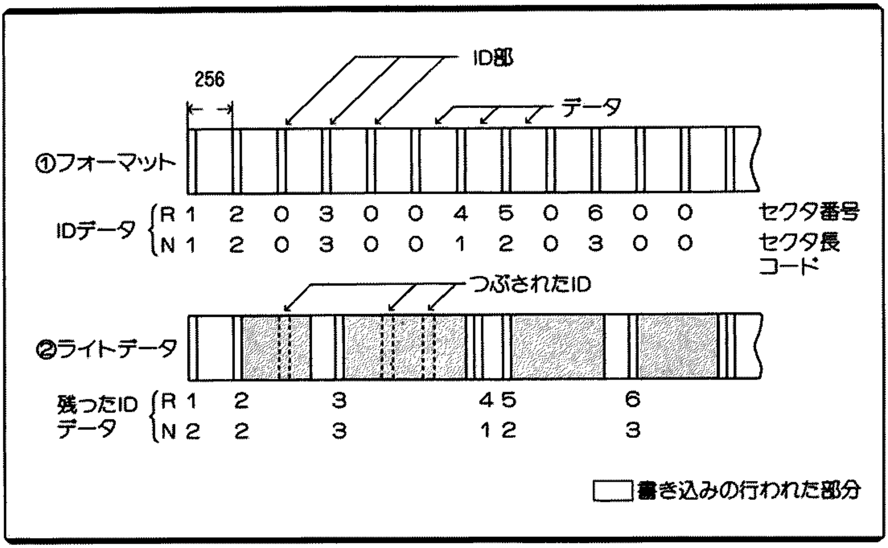
これはかなり異常なフォーマットです。フォーマット時に与えるIDデータ中のセクタ長コードは各セクタ様々な値をとることができますが、本当のセクタ長はフォーマット時に指定するセクタ長で、1トラック全体がフォーマットされるため、ひととおりに決まってしまいます。ですから1つのトラックのセクタはすべて同じセクタ長になるのが普通です。このフォーマットはその原則を破ったもので、なんと1トラック中に256、512、1024バイトなどのセクタ長をもつセクタを混在させようというものです。簡単にコピーできそうにないことは想像できるでしょう。

図2．1　1トラック中に異なるセクタ長が混在



これ以外の位置の場合は2回以上まわすことによって可能です。

図2．2　混在フォーマット作成の手順



◎ 作成方法

FDC（μPD765A）の制約により、フォーマット時のセクタ長はすべて等しくなります。しかし、各セクタのID中に書かれるセクタ長コードは任意の値をとることができます。では、そのセクタ長コードを変えたセクタに書き込みを行なうといったいどうなるでしょうか。セクタに対して書き込みが行なえるのは、指定したIDと一致するID部を持つセクタが発見されたときです。つまり、書き込み時に指定するセクタ長はフォーマット時のセクタ長ではなく、ID部に書かれているセクタ長なのです。書き込みはこのセクタ長に従って行なわれますから、書き込みを行なうことによって、ID部に書いたセクタ長コードに等しいセクタができあがるのです。1つ注意しなくてはいけないのは、セクタ長コードをフォーマット時のセクタ長よりも長くした場合で、書き込みによって次に続くセクタを破壊してしまうことです。例えば、256バイトでフォーマットしたセクタに対して、512バイトの書き込みを行なうと次に続く1セクタを、1024バイト

で書き込みを行なうと次に続く2セクタをつぶします。ですから、あらかじめつぶされる位置にはダミーのセクタを入れておきます。

図2．1のフォーマットは以下の手順で作成できます。

① 256バイトでフォーマットします。IDデータに注目。

② 512バイト（N＝2）、1024バイト（N＝3）のセクタに対して書き込みを行ないます。

プログラム例22

図2．1のフォーマットを実現します。まず、フォーマットを行なってから、セクタ番号2、3、5、6のセクタに書き込みます。

機械語

```
0000 B8715D      MOV AX,5D71H      ；フォーマットコマンド
0003 BB3000      MOV BX,48         ；12セクタ×4バイト
0006 B90901      MOV CX,0109H      ；セクタ長256バイト、トラック9
0009 BA0000      MOV DX,0000H      ；サーフェス0、初期データ00
000C 0E          PUSH CS           ；ES←CS
000D 07          POP ES
000E BD3900      MOV BP,OFFSET DATA
0011 CD01        INT 1B            ；BIOSコール
0013 B455        MOV AH,55H        ；ライトデータコマンド
0015 BB0002      MOV BX,200H       ；データ長512バイト
0018 B502        MOV CH,2          ；セクタ長512バイト
001A B202        MOV DL,2          ；セクタ番号＝2
001C BD0001      MOV BP,100H       ；書き込むデータの格納域アドレス
001F CD1B        INT 1BH           ；BIOSコール
0021 B455        MOV AH,55H        ；ライトデータコマンド
0023 B205        MOV DL,5          ；セクタ番号＝5
```

```
0025 CD1B                INT 1BH       ;BIOSコール
0027 B455                MOV AH,55H    ;ライトデータコマンド
0029 BB0004              MOV BX,400H   ;データ長 1024 バイト
002C B503                MOV CH,3      ;セクタ長 1024 バイト
002E B203                MOV DL,3      ;セクタ番号=3
0030 CD1B                INT 1BH       ;BIOSコール
0032 B455                MOV AH,55H    ;ライトデータコマンド
0034 B206                MOV DL,6      ;セクタ番号=6
0036 CD1B                INT 1BH       ;BIOSコール
0038 CF                  IRET

0039 09000101 DATA       DB 9,0,1,1    ;IDに書き込むデータ
003D 09000202            DB 9,0,2,2
0041 00000000            DB 0,0,0,0    ;ダミー
0045 09000303            DB 9,0,3,3
0049 00000000            DB 0,0,0,0    ;ダミー
004D 00000000            DB 0,0,0,0    ;ダミー
0051 09000401            DB 9,0,4,1
0055 09000502            DB 9,0,5,2
0059 00000000            DB 0,0,0,0    ;ダミー
005D 09000603            DB 9,0,6,3
0061 00000000            DB 0,0,0,0    ;ダミー
0065 00000000            DB 0,0,0,0    ;ダミー
                         END
```

◎ チェック方法

チェックは作成と違って簡単です。それぞれのセクタ長のセクタをすべて読んでみて、エラーがなくリードできることが確認されればOKです。

◎ コピー方法

まず、フォーマット時のN（セクタ長）を知らねばなりません。最も確実な方法はＩＤの間隔を計ることです。フォーマット時にはセクタ長は１種類ですから、ＩＤは等間隔にあるはずです。もし、途中のＩＤをつぶしたとしても間隔は元の２倍、３倍等になるだけですから、観測された間隔の最大公約数を求めればそれがフォーマット時の間隔となります。

ＩＤの間隔を計るには、リードダイアグノスティックで１トラック全体を読み出して、ＩＤらしきパターンを探すということも考えられますが、データ中に同じようなＩＤらしきパターンがあっても区別できないため、確実ではありません。ＩＤを間違いなく見つけることができるのはリードＩＤであり、これを使う以外に方法はありません。間隔はタイマを使って時間で計ります。間隔が求まれば同時にダミーセクタをいくつ入れたらよいかも求まりますから、あとは作成方法と同じです。

プログラム例２３

ＩＤの間隔を計ります。同時にＩＤも読み込んで表示します。時間単位は100μsecです。タイマの使い方については付録３を御覧ください。このプログラムでは読み込むＩＤの個数は指定せず、先頭から１回転に要する時間（200msec）続けて読み出します。

機械語

```
0000 0E          PUSH CS
0001 1F          POP DS
```

```
0002 2E8A0E0001      MOV CL,TRACK
0007 2E8A360101      MOV DH,SURFACE
000C B8713A          MOV AX,3A71H     ；単密度でリードＩＤ，頭出し
000F CD1B            INT 1BH
0011 0E              PUSH CS
0012 07              POP ES
0013 E83400          CALL TMRSET
0016 FC              CLD
0017 B300            MOV BL,0
0019 BF0301          MOV DI,OFFSET IDAREA
001C 2E8A360101 LOOP:   MOV DH,SURFACE
0021 B8716A          MOV AX,6A71H     ；倍密度でリードＩＤ
0024 CD1B            INT 1BH
0026 880D            MOV [DI],CL      ；ＩＤのＣを格納
0028 47              INC DI
0029 8835            MOV [DI],DH      ；ＩＤのＨを格納
002B 47              INC DI
002C 8815            MOV [DI],DL      ；ＩＤのＲを格納
002E 47              INC DI
002F 882D            MOV [DI],CH      ；ＩＤのＮを格納
0031 47              INC DI
0032 2EA19500        MOV AX,TIME      ；現在の時刻を読み出す
0036 AB              STOSW            ；時刻を格納
0037 FEC3            INC BL           ；セクタ数を数える
0039 3DD007          CMP AX,2000      ；時刻が 2000 を超えたか？
003C 72DE            JB LOOP
003E 2E881E0201      MOV KOSU,BL      ；セクタ数を格納
0043 E402            IN AL,02         ；割り込みを禁止
0045 0C01            OR AL,01         ；
0047 E602            OUT 02,AL        ；
```

```
0049 CF                 IRET
             ;
004A 1E          TMRSET:PUSH DS              ；インターバルタイマの初期化
                                             ；サブルーチン
004B 33C0               XOR AX,AX            ；DS←0
004D 8ED8               MOV DS,AX
004F FA                 CLI
0050 B88700             MOV AX,OFFSET TMRINT
0053 BE2000             MOV SI,0020H
0056 8904               MOV [SI],AX          ；ベクタテーブルの書き換え
0058 46                 INC SI
0059 46                 INC SI
005A 8CC8               MOV AX,CS
005C 8904               MOV [SI],AX
005E FB                 STI
005F 2EC70695000        MOV TIME,0           ；時刻←0
     00
0066 B034               MOV AL,34H           ；インターバルタイマセット
0068 E677               OUT 77H,AL
006A F606010580         TEST BYTE PTR .501H,80H；8MHZか調べる
006F B8C800             MOV AX,200
0072 7503               JNZ CLK8M            ；8MのときはCLK8Mへ
0074 B8F600             MOV AX,246
0077 E671        CLK8M:OUT 71H,AL            ；カウント値のセット
0079 8AC4               MOV AL,AH
007B E671               OUT 71H,AL
007D FA                 CLI
007E E402               IN AL,02             ；割り込み許可
0080 24FE               AND AL,0FEH
0082 E602               OUT 02,AL
```

```
0084 FB              STI
0085 1F              POP DS
0086 C3              RET
            ;
0087 50     TMRINT:  PUSH AX          ；割り込み処理ルーチン
                                      ；100μSECごとに呼ばれる
0088 2EFF069500      INC CS:TIME      ；時刻を１インクリメント
008D FA              CLI
008E B020            MOV AL,20H
0090 E600            OUT 00,AL
0092 FB              STI
0093 58              POP AX
0094 CF              IRET
            ;
0095 0000   TIME     DW 0             ；時刻を格納する
            ;
                     ORG 100H
0100        TRACK    RS 1             ；トラック番号
0101        SURFACE  RS 1             ；サーフェス番号
0102        KOSU     RS 1             ；読んだセクタ数
            ;
0103        IDAREA   RS 100           ；読んだＩＤ等を格納する領域
                     END
```

ＢＡＳＩＣ

```
10 INPUT"トラック番号=";TR: POKE &h100,TR
20 INPUT"サーフェス番号=";SUR: POKE &h101,SUR
30 A=0 : CALL A
40 KO=PEEK(&h102)
50 PRINT"Ｃ　Ｈ　Ｒ　Ｎ　前のＩＤとの間隔 (×100μSEC)"
```

```
60 FOR I=0 TO KO-1
70   FOR J=0 TO 3
80     PRINT PEEK(I * 6+J+&h103);
90   NEXT J
100   TT=PEEK(I * 6+&h107)+256 * PEEK(I * 6+&h108)
110   PRINT TT-T
120   T=TT
130 NEXT I
```

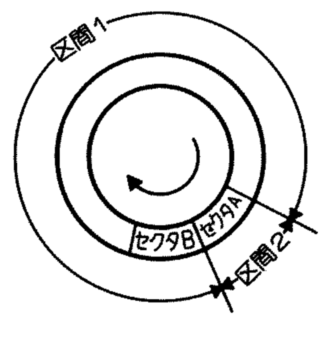


2.2 上級テクニック 1トラック中に同じIDが複数存在

◎ 原　　理

1つのトラック上のセクタを指定する唯一のキーは、ID部に書かれているセクタ番号です。では、1つのトラック上に同じセクタ番号をもつセクタが複数あったらどうなるでしょうか。FDCはリードコマンドを受け取ると現在のヘッドのある位置から探し始め、最初に出会った、指定したIDと一致するID部をもつセクタを読み込みます。例えば、下図でセクタA，Bは両方ともセクタ番号が2であったとします。もし、区間1をヘッドが通過しているときにリードコマンドを発すればセクタAが、区間2を通過しているときに発すればセクタBが読み込まれるわけです。

図2．3　同じIDが複数存在するとき



ですからこのフォーマットでは、ただリードを行なっただけではどのセクタのデータを読んだのかわからないのです。このため忠実にコピーすることも難しくなります。

◎作成方法

フォーマットだけならいとも簡単です。ＩＤに書き込むデータを同じにするだけですから。ところが、特定のセクタにデータを書き込もうとすると少し工夫がいります。例えば、全セクタが同じＩＤだとするとライトデータコマンドを発したときに、どこをヘッドが回っているかで、書き込まれるセクタが違ってくるからです。どこのセクタに書き込んでもいいのならこれでかまいませんが、ある特定のセクタに書きたいのなら、ライトコマンドを与えるタイミングは目的のセクタの直前のセクタを通過中のときに限られます。

そこでまず、ヘッドをトラックの先頭に位置づけ、トラックの先頭から必要な数のセクタをリードＩＤで読み飛ばします。目的のセクタがｎ番目なら、ｎ－１回リードＩＤを行なえばいいわけです。リードＩＤの直後、ヘッドは直前のセクタのＩＤ部直後にありますから、ここでライトコマンドを発すれば目的のセクタに書き込まれます。

プログラム例２４

すべてのセクタのＩＤを、同じにフォーマットしたトラックの、指定した１つのセクタのみに書き込みを行ないます。書き込むデータは２００Ｈ番地以降に用意されているものとします。セクタ長は256バイトとします。

機械語

```
0000 0E            PUSH CS
0001 1F            POP DS
0002 2E8A0E0001    MOV CL,TRACK
```

```
0007 2E8A360101      MOV DH,SURFACE
000C B8713A          MOV AX,3A71H    ；単密度でリードＩＤ，頭出し
000F CD1B            INT 1BH
0011 2E8A1E0201      MOV BL,KOSU     ；目的のセクタの先頭からの位置
0016 FECB            LOOP: DEC BL
0018 740C            JZ EXIT
001A 2E8A360101      MOV DH,SURFACE
001F B8716A          MOV AX,6A71H    ；リードＩＤで読み飛ばす
0022 CD1B            INT 1BH
0024 EBF0            JMPS LOOP
0026 B87145 EXIT:    MOV AX,4571H    ；ライトデータ、シークなし
0029 BB0001          MOV BX,100H
002C 2E8A0E0301      MOV CL,C
0031 2E8A360401      MOV DH,H
0036 2E8A160501      MOV DL,R
003B 2E8A2E0601      MOV CH,N
0040 BD0002          MOV BP,200H     ；書き込むデータの格納域
0043 CD1B            INT 1BH
0045 CF              IRET
                :
                     ORG 100H
0100        TRACK    RS 1            ；物理トラック
0101        SURFACE  RS 1            ；物理サーフェス
0102        KOSU     RS              ；目的のセクタの先頭からの位置
0103        C        RS 1            ；ＩＤのC
0104        H        RS 1            ；ＩＤのH
0105        R        RS 1            ；ＩＤのR
0106        N        RS 1            ；ＩＤのN
                     END
```

```
BASIC
10 INPUT"トラック番号=";TR: POKE &h100,TR
20 INPUT"サーフェス番号= ";SUR: POKE &h101,SUR
30 INPUT"書き込むセクタの位置（何番目）=";KO : POKE &h102,KO
40 INPUT"ＩＤのC=";C : POKE &h103,C
50 INPUT"ＩＤのH=";H : POKE &h104,H
60 INPUT"ＩＤのR=";R : POKE &h105,R
70 INPUT"ＩＤのN=";N : POKE &h106,N
80 A=0 : CALL A
```

◎ チェック方法

特定のセクタにしかるべきデータが書かれていることをチェックする、または、そのデータを読み出して利用することを考えます。作成方法と同じようにトラックの先頭からリードＩＤで読み飛ばしたのち、目的のセクタをリードする方法もありますが、リードダイアグノスティックを使えばさらに簡単になります。

リードダイアグはトラックの先頭からリードしますから、セクタ長を正しく指定してマルチセクタ機能を使えばトラック先頭から書かれている順番に従って、複数のセクタを同時に読み込めます。目的のセクタが何番目のセクタかはわかっていますから、その部分を利用すればいいでしょう。

◎ コピー方法

リードＩＤで調査の結果トラック中に同じＩＤがあることがわかったら、チェック方法に示したようにして、データを読み出し作成方法のようにして書き込めばいいでしょう。

2.3 上級テクニック GAP3の長さをチェック

◎ 原理

ＧＡＰ３とはセクタ間のギャップのことで、連続して続くセクタをスムーズに読み出せるように、また、データ部に書き込みを行なったときに若干の回転変動があっても吸収できるように、設けられているものです。ＧＡＰ３の長さは特に短くしないかぎり支障はなく普通の読み書きに関係ありません。ですからＧＡＰ３の長さがいくつであっても普通はまったく問題となりません。しかしこのプロテクトは、普通は問題とならないＧＡＰ３の長さをチェックするもので、いままでのプロテクトとは傾向が違います。

このプロテクトが施されたフロッピディスクは"backup.n88"でもエラーなくコピーできますが、いざ動かしてみると動かないということになります。エラーせずにコピーできるので、プロテクトがかかっているトラックがわかりにくく、そういう意味でかなり高度なプロテクトです。

◎ 作成方法

フォーマット時に、パラメータＧＰＬの値を標準と違う値にします。ＢＩＯＳではＧＰＬの値を変えることができないので、知識編３．３のようにするか、拡張ＢＩＯＳ（付録１）を使ってください。

プログラム例２５

ＧＡＰ３を標準より１バイト長くしてフォーマットします。セクタ長 256 バイトでは、標準のＧＡＰ３は 54 バイトですが、それを 55 バイトにします。ＢＩＯＳではＧＡＰ３を変えられないので拡張ＢＩＯＳを使います。トラックは 30、サーフェスは０とします。

機械語

```
0000 B8715D          MOV AX,5D71H     ；フォーマットコマンド
0003 BB4000          MOV BX,64        ；16 セクタ×４バイト
0006 B91E01          MOV CX,011EH     ；セクタ長 256 バイト、トラック 30
0009 BA1100          MOV DX,0011H     ；サーフェス０、初期データ１１Ｈ
000C 0E              PUSH CS
000D 07              POP ES
000E BD1A00          MOV BP,OFFSET DATA
0011 BE3710          MOV SI,1037H     ；セクタ数＝１０Ｈ（16）
                                      ；ＧＰＬ＝３７Ｈ（55）
0014 BFFFFF          MOV DI,0FFFFH    ；リセットを使わない
0017 CD40            INT 40H          ；拡張ＢＩＯＳコール
0019 CF              IRET
001A 1E000101DATA    DB 30,0,1,1      ；ＩＤに書き込むデータ
001E 1E000201        DB 30,0,2,1
0022 1E000301        DB 30,0,3,1
0026 1E000401        DB 30,0,4,1
002A 1E000501        DB 30,0,5,1
002E 1E000601        DB 30,0,6,1
0032 1E000701        DB 30,0,7,1
0036 1E000801        DB 30,0,8,1
003A 1E000901        DB 30,0,9,1
003E 1E000A01        DB 30,0,10,1
0042 1E000B01        DB 30,0,11,1
```

```
0046 1E000C01       DB 30,0,12,1
004A 1E000D01       DB 30,0,13,1
004E 1E000E01       DB 30,0,14,1
0052 1E000F01       DB 30,0,15,1
0056 1E001001       DB 30,0,16,1
                    END
```

◎チェック方法

ここでも、リードダイアグを使ってGAP 3の部分を読み込み、その長さを数えてから作成時に設定した値と一致するかをチェックします。ここで問題は、セクタにデータを書き込むと回転変動によってGAP 3の長さが変わることがあることで、正確にフォーマット時のGAP 3の長さを測定するには、データを書き込んでいないセクタを1つ以上残しておかなくてはなりません。データを書き込まないセクタはどこにあってもかまいません。

リードダイアグによってGAPを読む方法は、知識編2. 2の例題で説明したようにセクタ長よりも長いNを指定することによって可能となります。しかし、データを書き込んでいないセクタが先頭以外の場所にある場合は、リードダイアグで先頭から読むとビットずれの影響を受けますから、マルチセクタ機能を使って工夫してください。1セクタ 256バイト、GAP 3が標準に近い場合は次のようになります。

表2. 1　リードダイアグのマルチセクタ機能で読めるGAPの位置

Nの値	GAP3の位置（番目）
2（512バイト）	1，3，5，7，9，11，13………
3（1024バイト）	1，4，7，10，13，16………
4（2048バイト）	1，7，13，19………

GAP3さえ読み込めばあとは簡単です。データを書き込んでいないセクタではビットずれは起こりませんから、GAPの値は4Eです。GAP3はCRCの直後から次のセクタのSYNCの前までです。SYNCは00で区別できますから、CRCの直後から4Eがいくつ続いているか数えれば，正確に長さが求められます。

プログラム例26

4番目のGAP3の長さを計ります。4番目のセクタはフォーマットのみが行なわれているものとします。セクタ長は256バイトとします。4番目のGAP3を読むにはN=3にして2セクタ分（2048バイト）読むと、2セクタ目が実際の4番目のセクタに相当し、2セクタ目の258バイトより後が目的のGAP3です。トラック9、サーフェス0とします。

機械語

```
0000 B87152            MOV AX,5271H      ;リードダイアグ
0003 BB0008            MOV BX,800H       ;2048バイト
0006 B90903            MOV CX,0309H      ;セクタ長1024バイト、トラック9
0009 BA0000            MOV DX,0000H      ;サーフェス0
000C 0E                PUSH CS
000D 07                POP ES
000E BD0002            MOV BP,200H       ;200H番地以降をワークとして使う
0011 CD1B              INT 1BH
0013 BE0207            MOV SI,702H       ;目的のGAP3の先頭を指す
0016 FC                CLD
0017 803C4E     LOOP:  CMP BYTE PTR[SI],4EH
                                         ;SIで指される値が4Eか?
001A 46                INC SI
001B 74FA              JE LOOP           ;4EのときはLOOPへ
001D 81EE0307          SUB SI,703H       ;GAP3の長さの計算
0021 2E89360001        MOV CS:GAP3,SI    ;GAP3に格納
```

```
0026 CF             IRET

                    ORG 100H
0100 0000    GAP3   DW 0              ；求めたGAP３の長さを格納する
                    END
```

BASIC

```
10 A=0 : CALL A
20 PRINT"GAP３の長さ=";PEEK (&h100)
```

◎ コピー方法

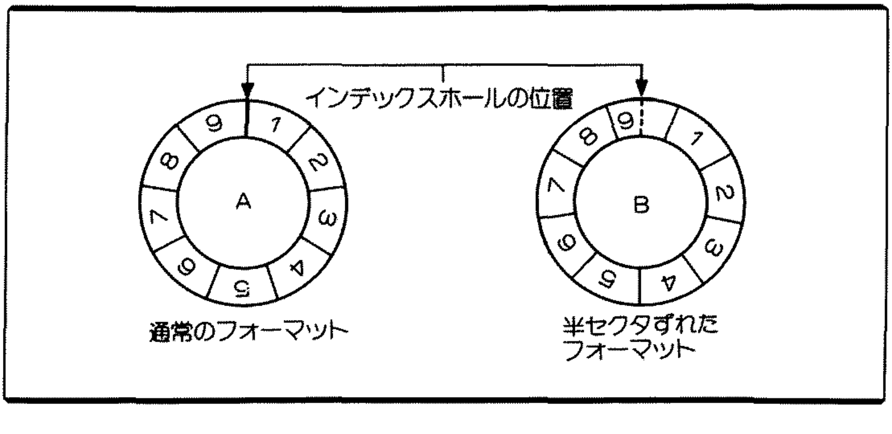
フォーマット時のGAP３の長さを求めなくてはなりません。チェック方法と同じようにすればいいのですが、今度は、どのセクタにデータが書かれていないのかがわかっていません。まず、それから調べます。第１の手がかりはフォーマットをのみしたセクタは、データがすべて同じ値であるということです。該当するセクタについてリードダイアグで続くGAP部を読んでみます（実はこれが少し難しい）。そこでビットずれを起こしていなければ、データが書き込まれていないセクタですから、GAP３を計ってください。

2.4 上級テクニック 先頭セクタの位置をずらす

◎ 原 理

これもエラーなくコピーできるプロテクトの一種です。下図のようにトラックの先頭を半セクタ分ずらしたものです。

図2．4 先頭セクタの位置をずらす



半セクタしかずれていないので、先頭のセクタは両方とも同じで、リードＩＤを行なってもまったく同じ結果となります。これでは、チェックもできないのではないか思いますがちゃんとできるのです。１つのトラックを読んでいる限りはわからなくても、２つトラックを比べれば違いはでてきます。裏面には正常な（つまりＡの）トラックを用意しておいて、表、裏のトラックを交互にリードＩＤするのです。表もＡのフォーマットなら両面とも同じ位置にＩＤがあるので１、２、

3、4、5．．．としか読み出されません。ところが表がBのフォーマットだとＩＤの位置が表裏で半セクタずつずれているため、両面のすべてのＩＤが読み出され、1、1、2、2、3、3、4、4．．．となるのです。

◎ 作成方法

裏のトラックには通常のフォーマットをするとして、表のトラックでセクタをずらす方法を考えます。

方法としてはフォーマットの時点でセクタをずらす方法と、データを書き込んでずらす方法があります。

1．　フォーマットだけでセクタをずらす方法

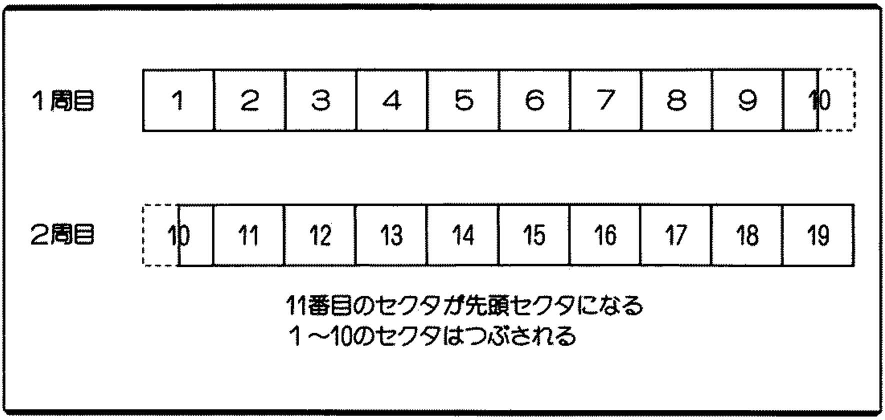
ＦＤＣはライトＩＤ命令を実行してフォーマットを行なうとき、必ずトラックの先頭からセクタを作っていきます。ですからこの命令によって、トラック先頭のセクタの位置をずらすなどできるはずがないと思うかもしれません。たしかに、ライトＩＤによって最初に書かれるセクタの位置は変えられませんが、トラックの先頭に来るセクタとは、必ずしも最初に書かれるセクタであるとは限りません。なにしろ、トラックは環状になっていますから、もうおわかりでしょう。ライトＩＤで書き込むセクタ数（ＳＣ）を多くすると、トラックを１周しても終了しないことを利用します。

一般に、ｉ番目のセクタのトラックの先頭からの位置（バイト）は、次式で表わされます。

倍密度の場合

位置＝｛146＋（62＋セクタ長＋ＧＡＰ３長）×（ｉ－１）｝mod １トラックの総バイト数 ……………………………… 式（２－１）

図2．5　フォーマットでＳＣを大きくすると



単密度の場合

位置＝｛73＋（33＋セクタ長＋ＧＡＰ３長）×（ｉ－１）｝mod 1トラックの総バイト数 ……………………………… 式（２－２）

半セクタ後方へずれるということは、この位置が

位置＝プリアンプルの長さ＋セクタ長／２

となればいいわけです。ｉだけを変えて先のフォーマット（512バイト／セクタ、ＧＡＰ３＝84（標準）、総バイト数＝6250）を実現するとしますと、ｉが11のときほぼこれを満たします。11番目のセクタが先頭セクタとなりますから、ＳＣは19にすればいいことがわかります。

プログラム例２７

１の方法で半セクタずらします。フォーマット時のセクタ数が標準より多いので、拡張ＢＩＯＳを使います。トラック40、サーフェス０とします。

機械語

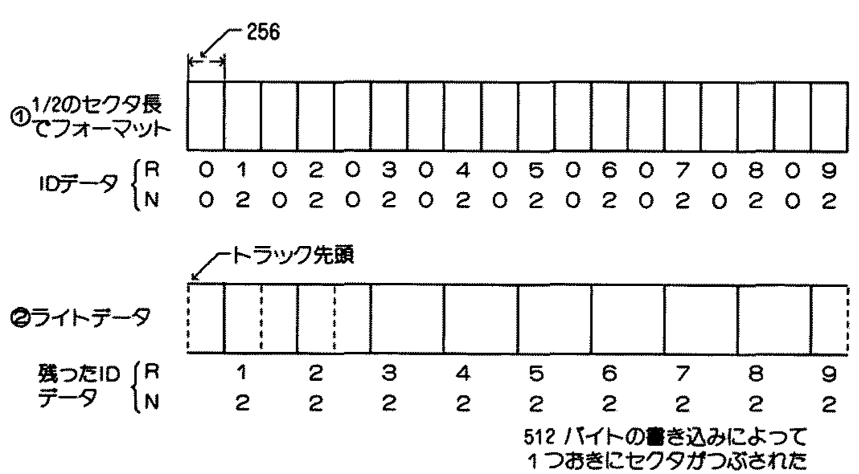
```
0000 B8715D          MOV AX,5D71H    ；フォーマットコマンド
0003 BB4C00          MOV BX,76       ；19セクタ×4バイト
0006 B92802          MOV CX,0228H    ；セクタ長512バイト、
                                     ；トラック＝28H（40）
0009 BA2200          MOV DX,0022H    ；サーフェス0、初期データ22H
000C 0E              PUSH CS
000D 07              POP ES
000E BD1A00          MOV BP,OFFSET DATA
0011 BE5413          MOV SI,1354H    ；セクタ数＝13H（19)、
                                     ；GPL＝54H（84）
0014 BFFFFF          MOV DI,0FFFFH   ；リセットは使わない
0017 CD40            INT 40H         ；拡張BIOSコール
0019 CF              IRET
                   :
001A 00000000 DATA DB 0,0,0,0        ；ダミー×10
001E 00000000        DB 0,0,0,0      ；2周目につぶされる
0022 00000000        DB 0,0,0,0      ；
0026 00000000        DB 0,0,0,0      ；
002A 00000000        DB 0,0,0,0
002E 00000000        DB 0,0,0,0
0032 00000000        DB 0,0,0,0
0036 00000000        DB 0,0,0,0
003A 00000000        DB 0,0,0,0
003E 00000000        DB 0,0,0,0
0042 28000102        DB 40,0,1,2     ；これが先頭セクタとなる
0046 28000202        DB 40,0,2,2
004A 28000302        DB 40,0,3,2
004E 28000402        DB 40,0,4,2
0052 28000502        DB 40,0,5,2
```

```
0056 28000602        DB 40,0,6,2
005A 28000702        DB 40,0,7,2
005E 28000802        DB 40,0,8,2
0062 28000902        DB 40,0,9,2
                     END
```

2. データを書き込んでセクタをずらす方法

異なるセクタ長の混在のところで説明したダミーセクタをつぶす方法を使います。トラックは環状ですから先頭セクタは最終セクタによってつぶせます。先頭セクタがつぶされたことによって、もともと2番目のセクタが新しい先頭セクタになります。ここでは、半セクタ分ずらしたいのですから、最初は2分の1のセクタ長でフォーマットすればよいでしょう。手順は次図のとおりです。

図2．6　データを書き込んで半セクタずらす



プログラム例２８

２の方法で半セクタずらします。セクタ長を２分の１、セクタ数を２倍にするとともにＧＡＰ３も短くなります。フォーマットには拡張ＢＩＯＳを使います。トラック40、サーフェス０とします。

機械語

```
0000 B8715D          MOV AX,5D71H     ；フォーマットコマンド
0003 BB4800          MOV BX,72        ；18セクタ×４バイト
0006 B92801          MOV CX,0128H     ；セクタ長256バイト、
                                      ；トラック＝２８Ｈ（40）
0009 BA2200          MOV DX,0022H     ；サーフェス０、初期データ２２Ｈ
000C 0E              PUSH CS
000D 07              POP ES
000E BD2700          MOV BP,OFFSET DATA
0011 BE2012          MOV SI,1220H     ；セクタ数＝１２Ｈ（18）、
                                      ；ＧＰＬ＝２０Ｈ（32）
0014 BFFFFF          MOV DI,0FFFFH    ；リセットは使わない
0017 CD40            INT 40H          ；拡張ＢＩＯＳコール
0019 B87155          MOV AX,5571H     ；ライトデータコマンド
001C BB0012          MOV BX,1200H     ；マルチセクタ機能を使って
                                      ；１から９まで１度に書き込む
001F B201            MOV DL,1         ；セクタ番号１
0021 BD0001          MOV BP,100H      ；書き込むデータの格納域
0024 CD1B            INT 1BH          ；ＢＩＯＳコール
0026 CF              IRET
                :
0027 00000000 DATA   DB 0,0,0,0       ；ダミー　ひとつおき
002B 28000102        DB 40,0,1,2
002F 00000000        DB 0,0,0,0       ；書き込みによってつぶされる
0033 28000202        DB 40,0,2,2
```

```
0037 00000000          DB 0,0,0,0
003B 28000302          DB 40,0,3,2
003F 00000000          DB 0,0,0,0
0043 28000402          DB 40,0,4,2
0047 00000000          DB 0,0,0,0
004B 28000502          DB 40,0,5,2
004F 00000000          DB 0,0,0,0
0053 28000602          DB 40,0,6,2
0057 00000000          DB 0,0,0,0
005B 28000702          DB 40,0,7,2
005F 00000000          DB 0,0,0,0
0063 28000802          DB 40,0,8,2
0067 00000000          DB 0,0,0,0
006B 28000902          DB 40,0,9,2
                       END
```

◎ チェック方法

原理のところで述べたように、ディスケットの表と裏の同じトラックを交互にリードＩＤして、両方のＩＤが、全部読めることをチェックします。

◎ コピー方法

まず、先頭セクタが通常よりずれていることを発見する必要があります。これにはタイマによる時間計測が最も確実です。トラックの頭出しをしてから、初めてＩＤが見つかるまでの時間を計るのです。回転速度はわかっていますから時間からだいたいのバイト数が求まります。それが標準でない場合には、それを満たすように式２－１、２－２から各パラメータ（特にｉ）を決定し、フォーマットしてください。

2.5 上級テクニック GAP3にデータを書き込む

◎ 原理

ＧＡＰ３とはセクタとセクタのすきまで、倍密度では通常４Ｅが、単密度ではＦＦが書き込まれています。しかし、この値に意味はなくＧＡＰ３にはどんな値が書かれていてもなんら影響はありません。そこで、ＧＡＰ３の一部に標準の値以外を書き込んでおいてそれをチェックしようというものです。セクタ数、セクタ長、ＧＡＰ３の長さなどすべて正常ですから、一見したところ異常を発見できないプロテクトとなります。

◎ 作成方法

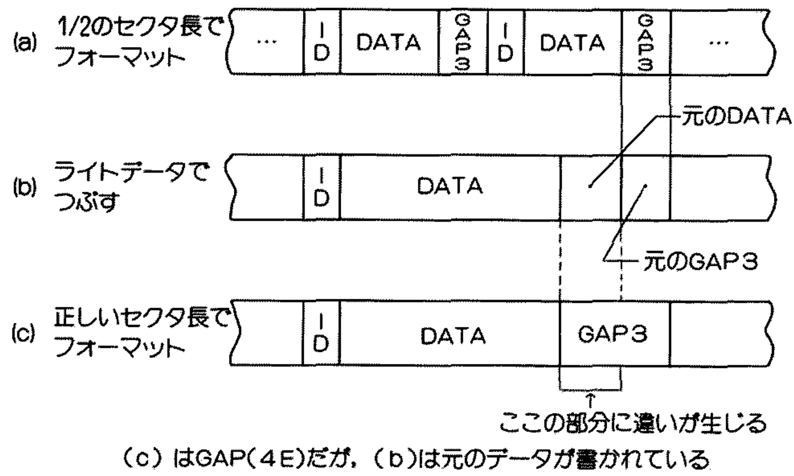
ダミーセクタをつぶすテクニックを使います。２分の１のセクタ長でフォーマットしておいて、１つおきにデータを書き込んでつぶします。そのようにして作られたセクタ図２．７（ｂ）と、始めから正しいセクタ長でフォーマットした（ｃ）ではＧＡＰ３部分で違いが発生するのです。（ｂ）には通常４Ｅのところに他の値が書かれています。

◎ チェック方法

違いの発生する部分を読み込んで調べます。普通はたとえビットずれがあっても、ＧＡＰ３はすべて何らかの同じ値となっているはずですから区別できます。

ＣＲＣ後、数バイト目から値を一定バイト数を調べすべて同じ値なら偽物です。

図２．７　ＧＡＰ３にデータを書き込む



◎ コピー方法

ＧＡＰ３におかしなデータが入っていないかどうかを調べます。チェック方法のようにしてＧＡＰ３に書かれている値を調べ、全部同じでない場合はこのプロテクトの可能性がありますから、作成方法のように処理してください。

2.6 上級テクニック DAMを破壊する

◎ 原　理

倍密度で128バイトのセクタ書き込みをするとそのセクタのDAM（データアドレスマーク）が破壊されます。DAMがないとそのセクタのデータが読めなくなるばかりでなく、先頭セクタのDAMを破壊すれば、リードダイアグノスティックまで使えなくなります。リードダイアグはフォーマット解析に非常に便利なコマンドですから、これが使えないとコピーツールにとっては大打撃を受けたことになります。先頭セクタにこのプロテクトを施し、他のセクタには他のプロテクトをかけておきます。するとリードダイアグが使えないため、他のプロテクトを調べることが困難になるのです。このプロテクトは他のプロテクトを守るためのプロテクトともいえます。

◎ 作成方法

先頭セクタを128バイトのセクタにしてフォーマットします。もちろん、トラック全体が128バイト／セクタである必要はないので、先頭セクタのIDのNを0にしておくだけです。次にそのセクタに書き込みをします。書き込みは正常に終了しますが、もう2度とそのセクタを読み出せません。他のセクタは普通のリードコマンドで読み出せます。

◎チェック方法

このプロテクトはコピーツールを困らせるのが目的ですから、特にプログラム中でチェックする必要はありません。注意することは、このプロテクトをかけるとリードダイアグが使えなくなるため、同じトラック中に施された他のプロテクトのチェックができなくなる可能性があることです。ただし、使えないのはリードダイアグだけですから、他のリード系コマンドでチェックできるようなプロテクトを施しておけばよいでしょう。

◎コピー方法

このプロテクト自体は簡単に再現できます。先頭セクタをリードしたときにMissing DAMエラー（リターンコード=F0）が出たらこのプロテクトです。あとは作成方法のようにすればいいでしょう。ところがこのプロテクトのために同じトラックに施された他のプロテクトが、解析しにくくなるのはどうしようもありません。リードダイアグ以外のコマンドを使ってがんばってください。

3.1 ビットずれの応用 データの書き込み有無をチェック

◎ 原 理

コピーといえば忠実にデータを読んで書き込むのが普通ですが、このプロテクトは、このコピーの基本を逆手にとってしまおうというものです。つまり、データの書き込みを行なうこと自体を禁止してしまうのです。でも、データを書き込まないでコピーできるはずはありません。ですから、ある特定のセクタに限って書き込みを禁止しておき、そのセクタにデータの書き込みが行なわれたかどうかをチェックするというものです。このテクニックは、フォーマットは正常であるためにプロテクトの方法を見つけることが難しく、かなり高度なプロテクトの1つです。

さて、ここでいう「データの書き込みを行なわない」とは、フォーマットだけがしてあって、一度もデータを書き込んでいない状態が保たれていることを意味します。フォーマットとは、1トラック全体に連続して初期データを書き込むことで、このときにID部、データ部が作成されます。データ部には初期データとして、単一のコードの繰り返しが書き込まれます。つまり、ある1バイトデータで埋めつくされる、ということです。

ですから、フォーマット後のセクタは、空になっているのではなく、データがちゃんと書かれているということになります。したがって、そのあと行なわれるセクタへのデータの書き込みは、データ部を新しいデータで書き変えることであって、けっして、はじめてデータを書き込むのではないことになります。

それでは、データを書き変えたかどうかを、どのようにして調べることができ

るのでしょうか。データの書き込みでは、データ部だけをもとのデータの上から重ね書きして書き変えますが、ドライブ装置の誤差のため、もとのデータ部とぴったり同じ位置にはならず、書き始め点、書き終わり点がずれてしまいます。このずれがあるかないかをチェックすることが、すなわち、データの書き込みが行なわれたかどうかのチェックになるのです。

◎ 作成方法

フォーマットのみをし、ある特定のセクタにはデータを書き込まないようにしておきます。ただそれだけのことで作成手順は単純ですが、コピーは単純にはいかないというタイプのプロテクトです。

ただ、念には念をというのなら、フォーマットのときに書き込まれる初期データとおなじデータで、もういちど、あるセクタを埋めなおすというのもよいでしょう。このセクタに関しては、ビットずれがあることによって本物であることを確認することになります。

◎ チェック方法

ビットずれは書き始めの点、および、書き終わりの点の２箇所にできますが、チェックのしやすから、書き終わり点のビットずれをチェックします。

書き終わり点は、ＤＡＴＡ部の最後にあるＣＲＣのあとのＧＡＰを１バイト書いたところです。ビットずれがある場合は、ＤＡＴＡ部から連続してリードダイアグでＧＡＰまでを読んだとき、続くＧＡＰの値が４Ｅでないことで確認することができます。

◎ コピー方法

どのセクタに、データの書き込みが行なわれていないかを、調べなくてはなり

ません。μＰＤ７６５Ａでフォーマットしたものであればフォーマット後いちども書き込みを行なわなければ、ＤＡＴＡ部が１バイトの同一データで埋められているはずです。したがって、同一データで埋められていないセクタは、そのこと自体、書き込みが行なわれたことを明らかに示していると考えてよいのですから、ビットずれをチェックする対象としては、同一データだけからなるセクタのみを考えればよいでしょう。

同一データからなるセクタであるからといって、それだけで書き込みがされていないと判断するのは早合点というものです。それらのセクタについて、リードダイアグで続くＧＡＰを読み４Ｅが続いているかどうかを調べます。４Ｅが続いているなら書き込みが行なわれていないセクタということになり、そのセクタには書き込みを行なわないようにします。

3.2 ビットずれの応用 ずれの度合いをチェック

◎ 原　理

データの書き込みによって前後にずれが生じることは先に述べました。そのずれを利用してデータの書き込みの有無をチェックできることも述べました。このプロテクトはデータを書き込んで積極的にずれを生じさせ、このずれ具合を計ろうというものです。時間的変動、または偶然誤差により観測されるずれは書き込みのたびに異なってきます。ですから、偶然にまったく同じずれになる可能性は低く、つまりは、同じものを作ろうとしても極めて困難で強力なプロテクトが実現できるのです。このプロテクトも作成方法は単純で、コピーが難しいというタイプのテクニックです。

◎ 作成方法

まず、普通にセクタにデータを書き込んでください。こんどは、これだけで終わりというわけにはいきません。3．1節では、ビットずれの有無だけをチェックすればよかったのに対し、ここでは、ずれの度合いをチェックしようというのですから、作成した正しいディスケットのどこかのセクタで、ビットずれの度合いを調べて覚えておかなければなりません。もちろん覚えておくというのは、その同じディスケットに書き込んでおくということに他なりません。

書き込みを行なったセクタでは、当然、書き終わり点でビットずれが生じているはずです。3．1節のチェック方法と同様に、リードダイアグでGAPまでを

読み込みます。倍密度ではもともとGAPに書き込まれた初期データは4Eですが、ビットずれの具合によってその値は変化してしまいます。

ずれの度合いと、変化したデータの値との関係を表3．1に示します。

表3．1　GAPのビットずれによる変化

ずれ(ビット)	0	0.5	1	1.5	2
値　(16進)	4E	21	9C	42	39

2.5	3	3.5	4	4.5	5
84	72	09	E4	12	C9

5.5	6	6.5	7	7.5	8
24	93	48	27	90	4E

8ビットずれるとちょうど1バイトでもとに戻り、ビットずれは外見上なかったことになります。変化した値は0.5ビットずれる毎に変わり、全部で16種類あります。GAPを読んで観測される値はこれらのうちのどれかですから、それを読み取って何ビットずれたかを求めることができるわけです。

チェックは同じ値が観測されるかどうかを調べるのですが、読み込み時の変動を考慮して±0.5ビットの余裕をみます。つまり、観測された値とその前後の値の、3つの値を覚えておくわけです。コピーしたディスケットでのビットずれが、その3つ値のいずれかになる確率は16分の3ですから、偶然に一致することもあり得ますが、複数のセクタにこのプロテクトを使用すればいくらでも確率を小さくすることはできます。

◎ チェック方法

リードダイアグにより目的のセクタのGAPを読み込み、そこの値が覚えておいたものと一致するかを調べます。オリジナルのディスケットのGAPのデータ

は、作成時にそのディスケットのどこかに書いておき、それをチェックルーチンで使います。

◎コピー方法

このプロテクトは、まったく普通に作成されたセクタを観測しているだけですから、プログラムを解読しない限りどこのセクタをチェックしているかを見つけることは不可能です。

したがって、プログラムを解読せずにコピーするためには、ディスク上のすべてのセクタを対象として、ＧＡＰの値をオリジナルのディスケットの値と一致するようにしなければなりません。つまり書き込みを行なって、偶然にビットずれが一致するまで何度もリトライする必要があるわけです。とにかく、非常に時間がかかります。

もちろん、プログラムを解読できれば、かなり処理は少なくなるかもしれません。しかし、プログラムの方も解読しにくいようにいろいろな工夫がこらされていることが多く、また、マイコンでプログラム解析ができるならともかく、人間がやらねばならないので、むしろ、時間はかかっても上の方法の方が楽かもしれません。

3.3 ビットずれの応用 不安定度のチェック

◎ 原理

かなり強力なコピーツールのひとつに、1トラックをまるごと読み込み、そのまま書き込むというハードウエアを用いたコピーツールがありますが、このプロテクトはそれらの弱点を突いたものです。どういう弱点かというと、そのようなコピーツールもたいていはサンプリングの周期が0.25ビットのずれを検知できるほど短くなっていないので、オリジナルのディスケットにあった不安定さをサンプリングの時点で吸収してしまい、忠実なコピーとはならない、ということなのです。

ずれの度合いがちょうど0.5ビットの整数倍になった場合を除いて、ビットずれは不安定さをもたらします。0.25+0.5Nビットの場合がもっとも顕著です。これはサンプリングのタイミングが、データビットとクロックビットのちょうど中間になるので、読み込み時のタイミングのちょっとした変動によって、どちらが読まれる（サンプリングされる）かが一定しないからです。

不安定なものをチェックするというとわかりにくいかも知れませんが、このプロテクトはこのような不安定さを持つかどうかをチェックするもので、実際の値をチェックするというものとは少々傾向が違っています。

また、ハードウェアによるコピーに対しては有効ですが、ソフトウェアによるコピーに対しては無力ともいえるので、単独ではあまり使われません。

◎ 作成方法

まったく普通にセクタに書き込みを行なってください。もちろん、別のプロテクトを組み合わせるなら、そのプロテクトの作成方法に従って書き込みを行なえばけっこうです。もしも、GAPのデータが安定した値を示すようなら、もういちど書き込みをやり直す必要があるでしょう。

◎ チェック方法

リードダイアグノスティックで1トラックをまるごと読み込み、GAP 4のところにあるデータを調べます。もともとは4Eですがビットずれのため表3．1のいずれかに変化しているはずです。1トラックまるごと読んでいるので、影響を受けるビットずれの箇所はセクタ数×2程度あり、最後の部分であるGAP 4の部分では、かなり不安定になることが予想されます。

10回程度連続して読み込めば、少なくとも1回は異なった値を示します。ところが、もし低いサンプリングレートでトラックをまるごとコピーしたとすると、0.5ビット以下のビットずれは保存されず、すべて0.5ビット単位となってしまうため、不安定さが消えてしまいます。ですから、何回読んでも同じ値を示すようなら、コピーされたものといえます。

プログラム例29

不安定性のチェックをします。安定している場合はこのプログラムから戻りません。チェックするトラックは50、サーフェス0です。

機械語

```
0000 B87152      MOV AX,5271H    ；リードダイアグ
0003 BB0020      MOV BX,2000H    ；8192バイト
0006 B93206      MOV CX,0632H    ；セクタ長8192バイト、
```

```
                                         ;トラック番号=32H (50)
0009 BA0000         MOV DX,0000H         ;サーフェス0
000C 0E             PUSH CS
000D 07             POP ES
000E BD0001         MOV BP,100H          ;100H番地以降をワークとする
0011 CD1B           INT 1BH              ;BIOSコール
0013 BE7017         MOV SI,6000          ;
0016 0E             PUSH CS
0017 1F             POP DS
0018 8B3C           MOV DI,[SI]          ;6000バイト目のデータをDIへ
001A B87152  LOOP:  MOV AX,5271H         ;リードダイアグ
001D CD1B           INT 1BH              ;BIOSコール
001F 393C           CMP [SI],DI          ;6000バイト目のデータが変化
                                         ;したかを調べる
0021 74F7           JE LOOP              ;同じならLOOPへ
0023 CF             IRET
                    END
```

◎ コピー方法

このプロテクトは、ハードウエアコピーツールを対象としたものですからFDCでコピーする場合には通用しません。ハードウエアコピーツールでこのプロテクトに対処するには、サンプリングレート（速度）を上げる、書き継ぎを行なうなどが考えられます。とにかく、不安定性が生じればよいということです。

4.1 回転変動の応用 トラック容量を測定する

◎ 原理

ディスクドライブの回転には変動があるということはすでに述べました。書き込みの周波数は一定ですから、回転がゆっくりであれば書けるデータは多くなり、速ければ少なくなります。±2%の回転変動があると1トラックに書けるデータ量の変動は2DDで±125バイトにもなります。読み込みのときは、この程度の回転変動には自動的にサンプリング周波数が追随するようになっているので、問題はありません。

トラック容量は、フォーマットのときのディスクドライブ装置の回転数に依存するわけですから、コピーしたディスケットのトラック容量は、一般にオリジナルと一致しないことになります。そこで、フォーマットしたときのトラック容量を計って覚えておき、実行時に、また計って同じかどうかチェックします。もし、コピーされたものであるとすると、偶然に同じトラック容量になっている可能性は非常に低いので、強力なプロテクトとなります。

◎ 作成方法

フォーマットしたあと、トラック容量を計るためにリードダイアグで1トラック全体を読み込みます。このとき、トラックを1周したところのデータは、ビットずれが生じていると安定して読めるとは限りません。そこで、万一ビットずれを起こしていても安定して読むことができ、しかもGAP、SYNCなどには現

われないデータの値を考えてみます。

それは2進で100100100100100…と0を2ビットはさんで続くデータで、このデータ列に限ってデータビットとクロックビットが同じ並びとなっています。16進表示では92 49 24 92で、この4バイトを第1セクタの先頭に目印として書いておきます。リードダイアグでトラック全体を読み込むといっても、実際にはトラック1周よりも長く読み込まなくてはなりません。そして、上記の目印データを探してトラック1周の長さを計るわけです。

目印データは、トラックのはじめの方から探すこともありませんから、2DDの場合で6100バイト目ぐらいからデータ92をサーチします。最初にみつけた位置がだいたいのトラック容量です。このトラック容量を、チェックルーチンのデータとしてディスクに書いて覚えておきます。この測定方法では、最大3バイトの誤差がでる可能性がありますが、チェックするときにはその分余裕を持たせることで解決できます。それでも、偶然に125分の3に入ることもありましょうが、可能性は低いので充分プロテクトとして通用します。

プログラム例３０

先頭セクタに先のデータを書き込んでトラック容量を測定します。測定に使用するトラックはあらかじめ256バイト／セクタの標準フォーマットがされているものとします。トラック9、サーフェス0を使います。

機械語

```
0000 B87155        MOV AX,5571H       ;ライトデータコマンド
0003 BB0001        MOV BX,100H        ;256バイト
0006 B90901        MOV CX,0109H       ;セクタ長256バイト、トラック9
0009 BA0100        MOV DX,0001H       ;サーフェス0、セクタ1
000C 0E            PUSH CS
000D 07            POP ES
000E BD3200        MOV BP,OFFSET DATA ;書き込むデータのアドレス
0011 CD1B          INT 1BH            ;BIOSコール
```

```
0013 B87152          MOV AX,5271H    ;リードダイアグコマンド
0016 BB0020          MOV BX,2000H    ;8192バイト
0019 BD0001          MOV BP,100H     ;100H以降に読み出す
001C CD1B            INT 1BH
001E BED418          MOV SI,6100+100H ;サーチ開始アドレス
0021 2E803C92 LOOP:CMP BYTE PTR CS:[SI],92H ;92Hを見つける
0025 46              INC SI
0026 75F9            JNE LOOP        ;見つからないときLOOPへ
0028 81EE0201        SUB SI,102H     ;トラック容量計算
002C 2E89363600      MOV CS:BYTES,SI ;BYTESへ格納
0031 CF              IRET
                :
0032 92492492 DATA DB 92H,49H,24H,92H
                :
0036 0000       BYTES DW 0              ;求めたトラック容量
                     END
```

BASIC

```
10 A=0 : CALL A
20 PRINT"トラック容量（バイト）=";PEEK (&h36)+256 * PEEK(&h37)
```

◎ チェック方法

作成方法と同様にしてもう一度トラック容量を計ります。その結果が、覚えておいたトラック容量の±3バイトに入るかどうかを調べます。入らなければコピーであると判断します。

コピー方法

このプロテクトはフォーマットがまったく正常なので、まず、解読しない限りこのプロテクトがかかっていることを発見できません。ですから、他のすべての手段をつくしても、まだコピーが動かないようなら、このプロテクトの可能性ありです。万一わかっても、このプロテクトは本物のディスケットを作成した時点の回転速度と同じ回転速度でないとコピーできないので、それを合わせる手段がない限りはコピーはできません。回転速度の違いを生じさせる最も大きな原因は個体差であり、本物を作成したドライブ装置と異なる限り難しいようです。

一つの方法は、ドライブ装置の時間的な変動を利用するもので、厳密には常に回転速度は変動しているので、何度もリトライしていればそのうちピッタリ合うこともあるだろうという、非常に原始的かつ消極的なもので、これは、自分のドライブ装置と本物のディスケットのトラック容量が近い場合に使える方法です。これに対し、もっとインテリジェンスの感じられる方法も考えられます。ドライブ装置についている調整ボリュームを操作して、強制的に回転数を合わせようというものです。ただし、機種によってボリュームの位置に手の届かないものもあり、本体を壊すことにもなりかねないので、あまりお勧めできません。また、この方法では、回転数をなんらかの手段によって監視しながら行なう必要がありますが、付録2でそのためのプログラムを紹介していますので御利用ください。

4.2 回転変動の応用 GAP3の長さを変える

原　　理

回転速度が変わると書き込めるデータの量が変わることはわかりました。つまり同じ長さのところに100バイト書いたり、102バイト書いたりできるわけです。　では逆に、同じデータ量を書くとするとどうなるのでしょう。書き込んだデータの占有する領域の長さが100ミリになったり、102ミリになったりするということになります。この長さを、直接、何ミリであるか測定するのは困難ですが、その長さの変動によりセクタのどこかの長さが何バイト分か短くなれば、それを調べることは容易にできることがおわかりでしょう。

このプロテクトは、フォーマットのときよりデータ書き込みのときの回転を速くすることによって、ＤＡＴＡ部が、続くＧＡＰ３を侵食し、結果としてＧＡＰ３の長さが短くなることを利用したものです。通常はＧＡＰ３の長さはどのセクタも同じですから、このような方法を使わない限り、特定のセクタだけのＧＡＰ３の長さを短くすることはできません。

作成方法

フォーマットのときとデータ書き込みのときで回転速度が異なっていなくてはなりません。一つのドライブの時間的変動に頼ってもできないことはありませんがあまり確実ではないので、回転速度がもとから違う２つのドライブを使うか、前節で述べたようにボリュームを調節して強制的に回転速度を変えます。

フォーマットのときはゆっくりと、データ書き込みのときは速くします。逆にするとGAP3の長さは長くなりますから、それも利用できることは、理解できるでしょう。セクタ長は長く設定するほど回転速度による長さの差が顕著になります。

すべてのセクタにデータを書き込んでもいいですが、回転速度がフォーマット時と同じものを、少なくとも1つは残してください。さもなければデータを書き込まないセクタを、1つは残しておきます。これは、もとのGAP3の長さと短くなったGAP3の長さを比較するためです。あるいは、もとのGAP3の長さをどこかに覚えていてもよいでしょう。

さらに、チェックのときのために、短くなったGAP3の長さも計っておき、どのくらい短くなったか、その長さの差を覚えておきます。

◎──────────── チェック方法

速い回転で書き込んだセクタのGAPと、フォーマット時と同じ回転で書き込んだ（または書き込みを行なっていない）セクタのGAPの長さを比較します。これは作成時にも行なって、あらかじめどの程度の差がでるかを計っておき、覚えておきます。

実行時のこの値が、あらかじめ計っておいた値と一致するかどうかを調べます。ビットずれが生ずるため1バイトの誤差を考慮して、±1バイト以上の差があればコピーしたディスケットであると判断します。GAPの長さの計り方は2．3節を参照してください。

◎──────────── コピー方法

このプロテクトは、1つのトラックを作成するのに複数の回転速度を必要とする複雑なプロテクトであるうえに、たとえ回転速度を自由に変えられたとしても、どのセクタをどの回転速度で書き込めばいいかを決めるのが困難なために、コピ

ーはなかなか大変です。

唯一の方法は、各セクタ毎にＧＡＰの長さを計って、同じ長さになるように回転速度を調整して、書き込みを行なうことですが、ＰＣ－９８００シリーズ本体だけでは、ほとんどコピーは不可能でしょう。



4.3 回転変動の応用 GAP3にごみを入れる

◎ 原理

通常GAP３はある一定の値、倍密度では４E、単密度ではFFで埋められますが、そこに回転速度の差を利用してそれら以外の値を入れようというものです。　前節で見たようにデータの書き込み時に回転速度を速くすると、GAPは短くなり、遅くすると長くなります。そこで同じセクタに２回の書き込みを、始めに速い回転で、次に遅い回転で行ないます。すると下図のように１回目で書き込んだデータの最後の部分が消されずに残ります。この部分をここではごみと呼んでいます。このごみが存在するかどうかをチェックするものです。

◎ 作成方法

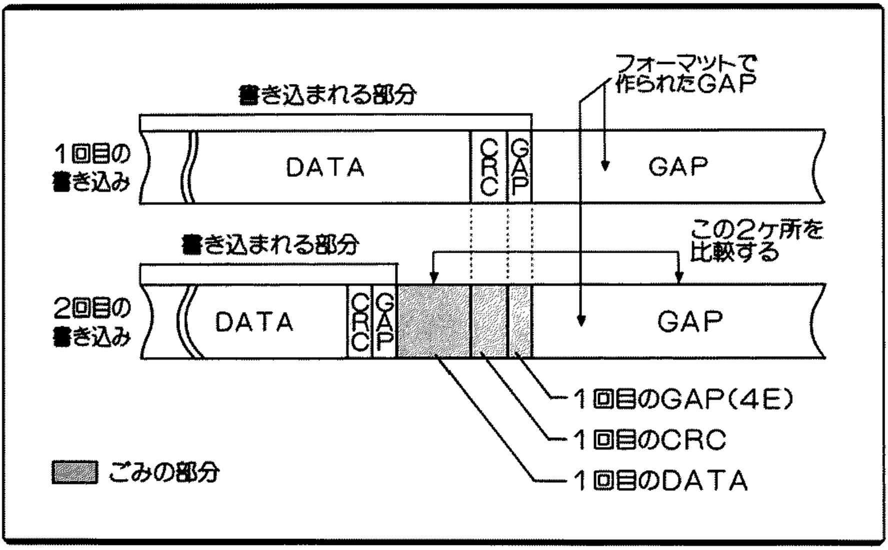
同じセクタに２回書き込みを、始めに速い回転で、次に遅い回転で行ないます。回転速度を変えるには前節と同じようにドライブ装置を代えるか、回転速度のボリュームで調節してください。セクタ長は長い方が差がでやすいのです。

◎ チェック方法

リードダイアグなどでGAP３の部分を読み込みます。データの書き込みでは、DATA、CRC、GAP１バイトが書き込まれますから、DATAの後４バイ

ト目から（たとえば1セクタ256バイトのときは、260バイト目から）数バイトがごみの部分のはずです。もしごみがなければGAPの部分は同じ値がずっと続くはずですから、ごみの部分と後のGAP部分の値を比較すればごみがあるかないかは判断できます。例えば、262バイト目と280バイト目を比較します。一致すればコピーされたものです。念のために作成時には、これが一致しないことを確認しておいてください。

図4．1　GAP3にごみを作る



◎ コピー方法

チェック方法と同様にすれば、このようなごみのあるセクタを検出することはできます。そのようなセクタに対しては作成方法と同じような処理を施してください。

4.4 回転変動の応用 普通では入らないセクタ数

◎ 原 理

ドライブの回転速度を変えるといろいろなプロテクトができます。このプロテクトは回転速度を遅くした応用です。回転速度を変えてしまっては本物まで読めなくなるのでは、と心配されるかもしれませんが、実は大丈夫なのです。フロッピディスクにはある程度の回転変動はつきものですから、それに対処するため、フロッピディスク装置は±2%程度の回転速度の違いなら問題なく読めるように保証されています（筆者の実験ではなんと20%でも大丈夫でした）。2%といっても1周が約10400バイト（8インチの場合）なら208バイトになるわけで、かなりの量です。2%回転を遅くすれば208バイト余分に書けるわけです。これだけ増えれば通常では入らない数のセクタを1トラックに入れることもでき、このように作られたフロッピディスクは、回転速度を変えない限りコピーできません。

◎ 作成方法

一つの例を挙げます。最もたくさんのセクタを入れるため、倍密度で1セクタを128バイトとし、GAP3は1バイトとします。ボリュームを調節するなどして、回転速度をできるだけ遅くしてフォーマットします。少し残酷ですがディスケットを上から押さえつけるという方法（知識編4．3）もあります。DATA以外の部分は1セクタにつき63バイトとなりますから、1セクタは63+128で、

191バイトです。1周を10600バイトとしても（8インチまたは5インチ2HDの場合）楽に55セクタは入ります。がんばれば56も可能でしょう。2DDなら、1周を6375バイトとして33セクタは入るはずです。

表4．1　1トラックに入るセクタ数の限界
(通常時と回転を2％遅くした時)

セクタ長（バイト）	5インチ2DD		2HDor8インチ	
	通　常	2%遅	通　常	2%遅
128	32	33	54	55
256	19	20	32	33
512	10	11	18	18
1024	5	5	9	9
2048	2	3	4	5
4096	1	1	2	2
8192	0	0	1	1

◎ チェック方法

すべてのセクタが正常に読み出せるかをどうかをチェックします。オリジナルのディスケットなら読み出せるはずですが、コピーならば、すべてのセクタが存在していないために、読み出しようのないセクタが出てくるはずです。

◎ コピー方法

リードID等でセクタ数を調べて、通常に入る数でなかったら回転を遅くする必要があります。IDだけのセクタの場合はこれよりはるかに多くすることもできますから、データを読み出してみて正常なセクタばかりである場合が、このプロテクトとなります。

5.1 リセットの応用 データがあるのにCRCエラー

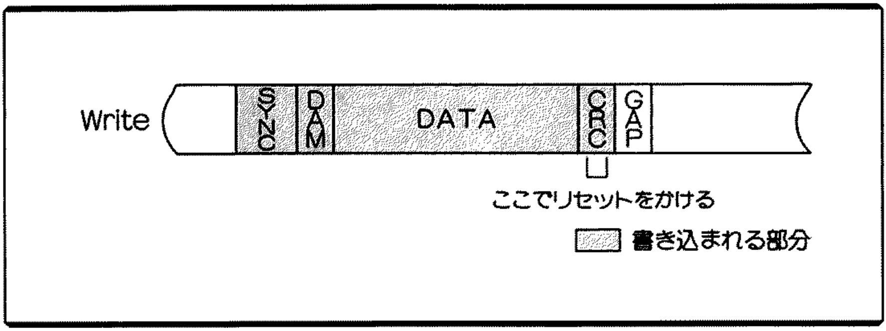
◎ 原理

データを読み込んだときにそれが正しく読まれたかどうかをチェックするため、ライトコマンドなどでデータを書き込むときには、DATAの全バイトから計算される2バイトのコード（CRC）を、データの直後に記録しておき、読み込み時にも同様の計算を行ないCRCが一致するかどうかを調べます。これが一致しない場合をCRCエラーと呼びます。普通、CRCエラーを起こすのはフォーマット時に実際と異なるNをIDに書き込んだ場合のみですが、その場合でもそのセクタにライトなどでデータを書き込んでしまうと、IDのNに従ってDATAおよびCRCが書き直されるので、CRCエラーは起こらなくなります。ですからデータが書いてあって、しかもCRCエラーを起こすことは、ディスケットが傷んだ場合以外有り得ないことですが、リセットを使ってデータの書き込み中、CRCを書き終わる前にFDCを止めればこれを実現できます。

◎ 作成方法

ライトデータまたはライトディリーテッドデータで書き込み中にリセットをかけます。リセットをかけるタイミングは、データを1バイト以上書き始めてからCRCを書き終わる前までなら、いつでもかまいません。

図5．1　リセットをかけるタイミング



なかでもいちばんコピーが困難と思われるのはＣＲＣを書き込んでいる間にリセットをかけた場合で、データはすべてが完全に書き込まれていますからそのデータを使用することができます。リセットを使用するには付録の拡張ＢＩＯＳを使ってください。

プログラム例３１

図５．１に示したようにデータのＣＲＣを書き込み中にリセットをかけます。拡張ＢＩＯＳを使います。全データを書き込んでからリセットをかけたいのでＢＸはセクタ長に等しくします。ＦＤＣは実際の書き込みより２～３バイト早く転送をしていますから、転送が終わった時点ではまだデータの書き込みは終了しておらず、ＣＲＣのタイミングに合わせるにはＤＩを３～４にするとよいでしょう。セクタ長256バイト、トラック９、サーフェス０、セクタ１とします。書き込むデータは１００Ｈ番地以降にあるとします。

機械語

```
0000 B87155      MOV AX,5571H    ；ライトデータコマンド
0003 BB0001      MOV BX,100H     ；データ長256バイト
0006 B90901      MOV CX,0109H    ；
0009 BA0100      MOV DX,0001H
000C 0E          PUSH CS
000D 07          POP ES
```

```
000E BD0001      MOV BP,100H     ;データのアドレス
0011 BE1010      MOV SI,1010H    ;セクタ数=16、GPL=16
0014 BF0300      MOV DI,3        ;データ転送後からリセットまでの
                                 ;時間、1単位=約1バイト
0017 CD40        INT 40H         ;拡張BIOSコール
0019 CF          IRET
```

◎ チェック方法

リード等でデータを読んでCRCエラー（リターンコード=B0）が出ることを確認します。リセットをかけたタイミングまでチェックするには、上で述べたようにCRC書き込み中にリセットをかけておきます。また、データの最後の1バイトを覚えておいて一致するかをみます。もし、コピーツールがリセットを使ったとしても、データの途中でリセットをかけていたら、最後のバイトは一致せずコピーと判定できます。

◎ コピー方法

データの読み込みでCRCエラーを検出し、かつデータの内容が一定でない場合は、リセットを使ったと思われます。しかし、リセットのタイミングを判定するのは難しく、このプロテクトのようにそれまでチェックする場合には、コピーはほとんど不可能でしょう。

5.2 リセットの応用 IDのCRCエラー

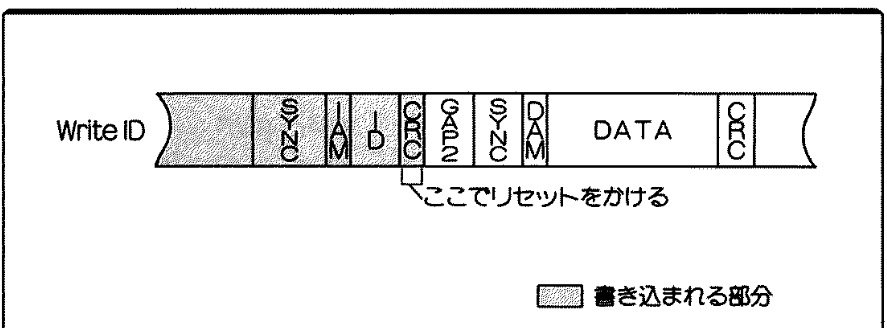
◎ 原　理

ＩＤにもデータと同じようにＣＲＣがあります。ＩＤのＣＲＣはフォーマット時に書かれるだけで普通はエラーにはなりません。リセットをかけることによって初めてＩＤのＣＲＣエラーを起こさせることができます。このエラーがあるとリードＩＤが効かなくなるので、そのセクタの存在すらわからなくなります。

◎ 作成方法

ライトＩＤでフォーマット中、ＩＤのＣＲＣ書き込み中にリセットをかけます。

図５．２　リセットをかけるタイミング



リードＩＤで発見できないように、他に正常なＩＤを持つセクタも作っておきます。リードＩＤはＩＤが正常である場合にのみＩＤを読み取るコマンドなので、

CRCエラーがあるとIDすら検出できなくなります。しかし、トラック上のすべてのセクタにこのエラーがあるとリードIDのリターンコードはC0となり、アンフォーマットの場合のE0と異なるため、IDにCRCエラーのあることがわかってしまうからです。

プログラム例32

図5．2に示したようにIDのCRC書き込み中にリセットをかけます。拡張BIOSを使います。IDのデータ書き込み直後にリセットをかけたいのですが、FDCは実際の書き込みより2～3バイト早く転送をしていますから、DIを3～4とします。BXは処理をしたいセクタの先頭からの位置×4とします。この例では、2番目のセクタを対象としますのでBXは8となります。トラックは9、サーフェスは0です。

機械語

```
0000 B8715D            MOV AX,5D71H       ；フォーマットコマンド
0003 BB0800            MOV BX,8           ；2セクタ×4バイト
0006 B90901            MOV CX,0109H
0009 BA2200            MOV DX,0022H
000C 0E                PUSH CS
000D 07                POP ES
000E BD1A00            MOV BP,OFFSET DATA
0011 BE3610            MOV SI,1036H       ；SC=16, GPL=54
0014 BF0300            MOV DI,3           ；データ転送後からリセットまでの
                                          ；時間、1単位=約1バイト
0017 CD40              INT 40H            ；拡張BIOSコール
0019 CF                IRET
                 ：
001A 09000101 DATA DB 9,0,1,1             ；IDに書き込むデータ
001E 09000201      DB 9,0,2,1
                   END
```

◎ チェック方法

このエラーを発生させるには正しいＩＤを指定してリードするしかありません。リターンコードはＡ０です。ＩＤを指定しなくてはいけないので、フォーマット時に書き込んだＩＤは覚えておく必要があります。また、ＩＤは完全に書き込まれていないと困りますので、リセットをかけるタイミングがＣＲＣ書き込み中に限定されるのです。

◎ コピー方法

このプロテクトが強いのは、まず、このようなプロテクトのかかったセクタの存在すらわからないことで、たとえわかって同じようにフォーマットしたくても、書き込むべきＩＤがわからないことです。唯一ＩＤを読むコマンドのリードＩＤが効かないのですから、どうしようもありません。

このプロテクトに対処する方法があるとすれば、リードダイアグで全体を読み込み、ビットずれを考慮しながらＩＤを探すことですが、これも先頭セクタに細工をして、リードダイアグをできなくすることによって、完全にお手あげとなります。

5.3 リセットの応用 DAMを破壊する

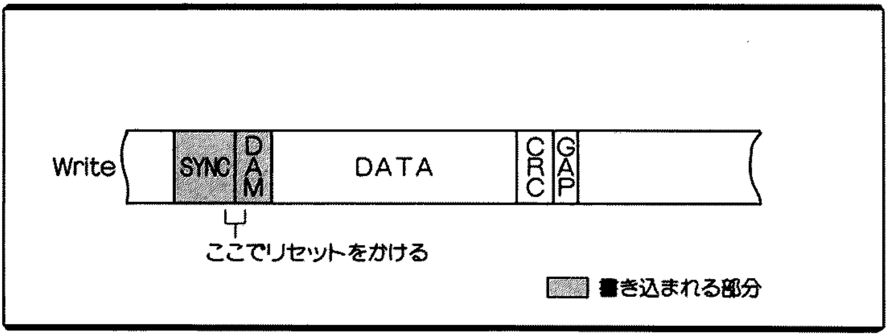
◎ 原　理

倍密度で128バイトのセクタに書き込みを行なうとDAMが破壊されることはすでに述べました。リセットを使えば128バイトに限らずどんなセクタでもDAMを破壊することができます。先頭セクタのDAMを破壊するとリードダイアグができなくなるという効果もあります。

◎ 作成方法

フォーマット時、データ書き込み時のどちらでも可能です。DAMを書き込み中にリセットをかければOKです。2．6節の方法と違いを出すためセクタ長は128バイト以外がいいでしょう。

図5．3 リセットのタイミング



プログラム例３３

データ書き込み時にＤＡＭを破壊します。データ転送直前にリセットをかけるため、ＤＩを０、ＢＸを１とします。ＦＤＣは実際の書き込みより、２〜３バイト程度早くデータを転送するので１バイトデータを転送した時点で、ちょうどＤＡＭを書き込むタイミングとなります。対象とするセクタはトラック５、サーフェス０、セクタ４で、すでにフォーマットはされているものとします。

機械語

```
0000 B87155      MOV AX,5571H     ；ライトデータコマンド
0003 BB0100      MOV BX,1         ；１バイト転送後、リセット起動
0006 B90501      MOV CX,0105H
0009 BA0400      MOV DX,0004H
000C 0E          PUSH CS
000D 07          POP ES
000E BD0001      MOV BP,100H
0011 BE1010      MOV SI,1010H     ；SC=16、GPL=16
0014 BF0000      MOV DI,0         ；転送終了後、即リセット
0017 CD40        INT 40H
0019 CF          IRET
```

◎チェック方法

リードを行なってリターンコードがＦ０となることを確認します。２．６節の方法では実現できないようにリードＩＤを行なって、Ｎが０でないことも確認します。

◎コピー方法

リードしたときにリターンコードＦ０のエラーがでたセクタについて、作成方法のようにしてＤＡＭを破壊します。

5.4 リセットの応用 GAP3にデータを書き込む

◎ 原理

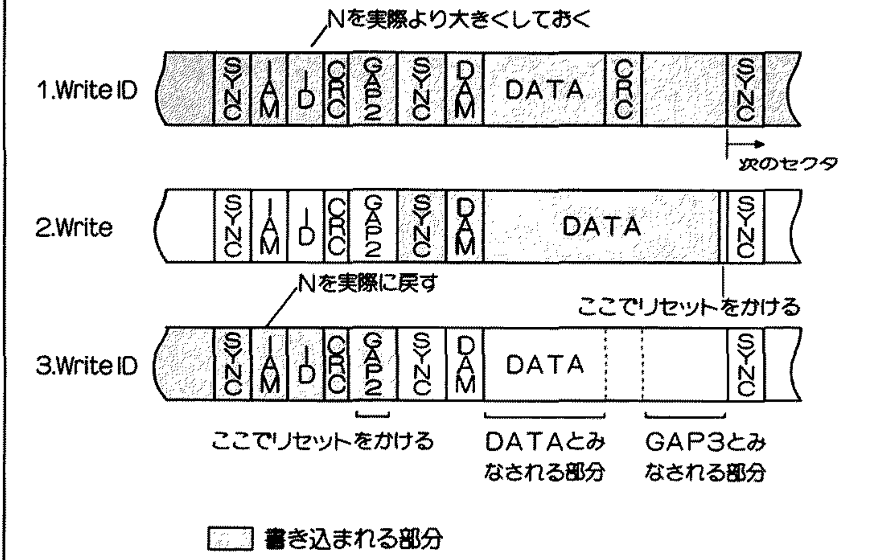
フォーマット時にGAP３に書き込まれるのは４Eですが、実は何が書いてあってもよく、このプロテクトはGAP３を、必要なデータ等を書いておく領域として扱ってしまうものです。GAP３に４E以外のデータを書くだけならリセットなしでもできますが（２．５参照）、信頼性を要求するとなるとビットずれのため一般には無理です。そこで、DATAの先頭からGAP３の領域までいっきに書き込むことを考えます。そのためには、始めはIDのNは充分に大きくしておく必要があるでしょう。リセットを使えば書き込みの途中で止めることはできますが、Nが小さければGAP３の領域まで書き込めません。GAP３までデータを書き込んだところで今度はフォーマットをしなおして、ID部まで書き込んでGAP２の途中でリセットをかけます。そのときIDのNを小さくするのです。

ここで、重要な点はID部を後から書き直すということで、この応用としては他にも倍密度でN＝０のセクタにデータを書くことなどが可能となります。

◎ 作成方法

原理で述べたように行ないます。手順を次ページの図に示します。

図５．４　ＧＡＰ３にデータを書く



ＧＡＰ２の長さはあまり変動が許されないので、回転変動がある場合には特に注意が必要です。必ずリードを行ないデータ部が読めることを確認してください。ＣＲＣの計算をして、ちょうどＣＲＣに相当するところに正しいＣＲＣを書いておけば、ＣＲＣエラーを出さなくすることもできます。そうしておけば、まず、気づかれません。せいぜい２．６節の方法と思われるだけです。

◎チェック方法

リードダイアグでＮを大きくしてデータ部から続けてＧＡＰまで読み込みます。ビットずれはありませんので、すべて正しいデータとして使えます。プロテクトのチェックにはこのＧＡＰ部分のデータを一部覚えておき、一致するかをみるといいでしょう。

◎ コピー方法

ＣＲＣエラーが出ないように処理されていると、このプロテクトがかかっていることに気づきません。発見するためには、ＧＡＰ２の中にビットずれがあるかどうかを調べる方法はあります。



5.5 リセットの応用 IDだけのセクタ

◎ 原　理

ＩＤ部までを書いたところでリセットをかけることによってＩＤ部だけのセクタを作ることができます。ＩＤ部だけだと22バイトで済むので、うまく書き込むと何と１トラックに２ＤＤで275セクタ、２ＨＤまたは８インチで468セクタ入ります。

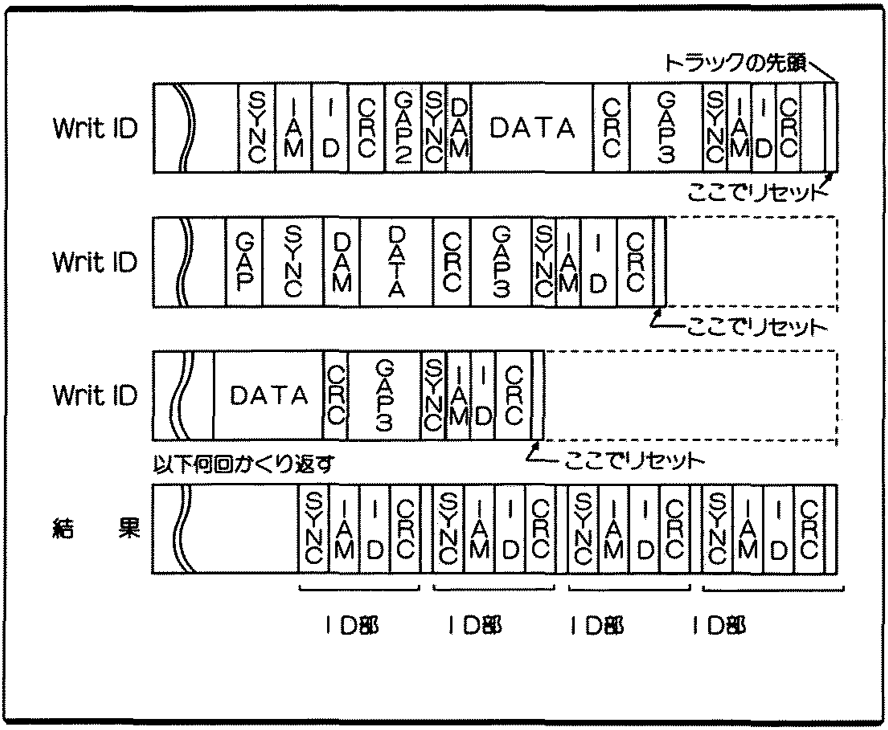
◎ 作成方法

ＩＤ部まで書いたところでリセットをかければよいのですが、１トラック全体をＩＤ部だけのセクタで埋めるとなると少し工夫が要ります。ライトＩＤは常にトラックの先頭から書き始めるのであって、任意の位置からは書き始められないからです。そこで、次のように止める位置を少しずつ早くしながら何回も繰り返します。書き込むＩＤはすべて異なるようにするとよいでしょう。

◎ チェック方法

リードＩＤをセクタ数分行なって同じだけのセクタがあることを確認します。

図5．5　1トラックをＩＤ部だけのセクタで埋める



◎ コピー方法

リードＩＤを行なえば異常にセクタが多いことはわかりますから、作成方法と同じように行ないます。

5.6 リセットの応用 単密度、倍密度の混在

◎原理

ＦＤＣは単密度、倍密度のどちらでも書くことはできますが、普通は１トラック内では記録密度を変えることはできません。それは、ライトＩＤが必ず１周全体に書き込みを行なってしまうからで、リセットを使って途中で止めればそれも可能となります。

◎作成方法

まず単密度か倍密度で１周をフォーマットします。次に密度を変えてもう一度フォーマットしますが、途中でリセットをかけます。また、さらに密度を変えてもう一度フォーマットし、２回目より早くリセットをかけます。これを繰り返すことによってかなり複雑な単密度、倍密度のサンドイッチができあがります。

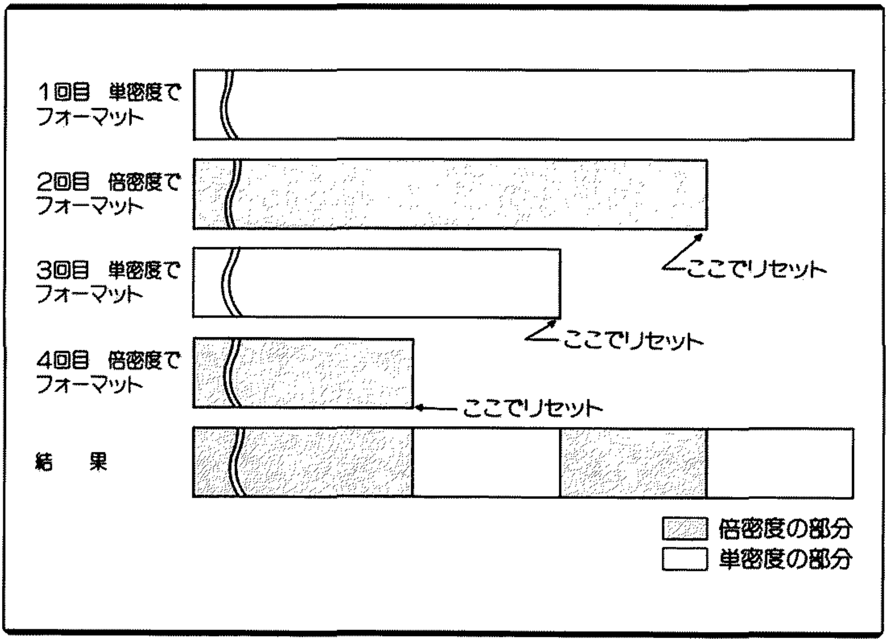
◎チェック方法

それぞれの密度でリードＩＤまたはリードを行ない、全セクタが正しく読み出されることを確認します。

◎ コピー方法

両方の密度でリードＩＤを行なってセクタが発見されたらこのプロテクトです。まず、それぞれの密度でリードＩＤで時間を計って、各セクタの位置を求めます。両方の結果を突き合わせて、どの位置にどの密度のセクタがあるかを推測し、後はそれを実現するように最後のセクタから順番に書いていきます。

図５．６　単密度、倍密度を混在させる



単密度、倍密度、それぞれのセクタにデータを書き込むことも可能です。

5.7 リセットの応用 アンフォーマットのデータを読む

◎ 原　理

アンフォーマットとは1トラックに1つもセクタがないことですが、セクタが1つもないとリードダイアグすら使えないので、まったくデータが読めず何が書かれているかわかりません。しかし一般に、アンフォーマットにするときには、1周以上データを書き込んでIDをつぶすのですから、なんらかのデータは書かれているわけです。そこで、それを読むために、リセットを使ってIDとDAMだけを作ってやります。フォーマットして、先頭セクタのIDとDAMを書いたところでリセットをかければいいわけです。この方法ではビットずれは起こしますが、書いたデータかどうかは判定できないわけではなく、強力なプロテクトとなります。

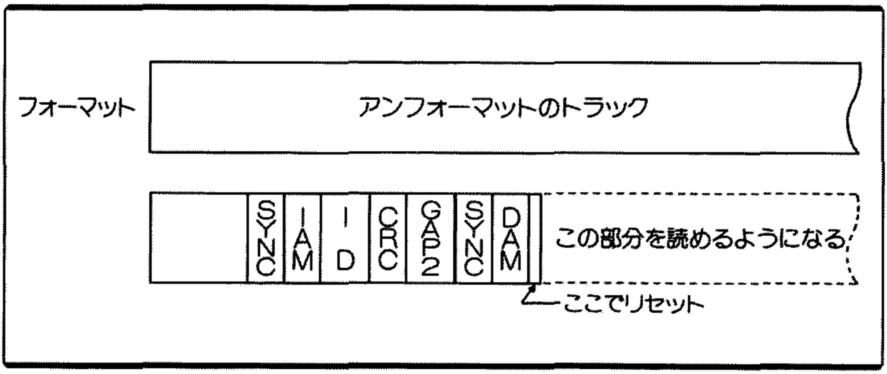
◎ 作成方法

後にデータをチェックすることを考えてフォーマットではなく、データ書き込みによってIDをつぶし、アンフォーマットを作ります。そこで書き込むデータは、ビットずれの影響を受けにくいものがよく、00,FF いずれかまたは、92,49,24 の連続などです。または、これらを何バイトか毎に交互に繰り返してもいいでしょう。

◎ チェック方法

まず、アンフォーマットであることを確認します。次に、テストするトラックの先頭部分をフォーマットしなおし、ID部およびDAMを作ります（このときにリセットを使います。でないとデータを全部つぶしてしまいます）。作ったIDを使ってリードダイアグし、何が書かれているかを読み取ります。ビットずれは起こりますが、作成方法に示したようにわかりやすいパターンを書き込んでおけば、本物かどうかのチェックはできます。例えば、200バイト目は00かFF、1000バイト目は92か42か24のいずれかでなくてはならない、というチェックです。

図5．7 リセットのタイミング



この方法はフォーマットを伴うので新しくセクタができてしまいます。チェックの結果OKなら作成方法に従って、もういちどアンフォーマットに戻しておきます。コピーと判断された場合は普通のアンフォーマットにしてしまいます。このようにアンフォーマットに戻しておくのは、ここでチェックしていることがばれないようにするためです。

◎ コピー方法

アンフォーマットの場合データが読めないのですからどうしようもありません。このプロテクトに対処するにはチェック方法のようにフォーマットしなおし、データを読んでみるしかありません。



6.1 ハングアップの応用 コピーツールをハングアップ

◎ 原理

知識編2．4節で述べたように、μＰＤ７６５Ａはある条件のもとでハングアップを起こすことがわかっています。その条件をもう一度示しておきましょう。

① 倍密度で読んだときすべてのデータビットが０であり、それが永遠につづくこと。

② 倍密度で読んだときクロックビットの並びの中に、11111010111111という並びが含まれること。

これらの条件を満たすための具体的な方法は、

① 単密度でフォーマットし、

② 初期データとしてＥ７，ＣＦ，９Ｆ，３Ｆ，７Ｅ，ＦＣ，Ｆ９，Ｆ３のどれかの値を使うこと

でした。

また、ハングアップをプロテクトに使うためには、単にハングアップさせるというだけでなく、単密度でのリードＩＤも正常に行なえないようにしておかなくてはならないことも知識編で説明したとおりです。したがって、単密度で、アンフォーマットに見せかけるように、フォーマットして、リードＩＤを失敗させ、倍密度でリードＩＤを行なわせるようにしむけ、そこでハングアップさせる、という方法になります。

◎ 作成方法

上の条件の①を満足するためには、単密度でフォーマットしたとしても、トラックの先頭部でのデータの継ぎ目が問題になります。すなわち、データビットとクロックビットの交互の繰り返しが、その継ぎ目でうまく続かなければならないということです。これは、ディスクドライブの回転むらに左右されることですから、まったく偶然にたよるしかなく、ハングアップするようになるまで、何度かフォーマッティングをリトライしなければなりません。

フォーマッティングの仕方は、単密度で普通にしてもよいのですが、単密度でのリードＩＤを失敗させるために、前に使ったアンフォーマットにみせるテクニックを使います。つまり、セクタ長をトラック１周を超えるように設定してフォーマットします。もちろん、初期データとしては、前に示した値のどれかを指定します。

プログラム例３４

倍密度で読んだときハングアップするようにフォーマットします。単密度でフォーマット、倍密度でリードを繰り返し、ハングアップするまでリトライします。単密度でフォーマットするときN＝５にしてアンフォーマットにします。このプログラムを動かすと画面に"TRYING n TIMES"とリトライ回数を表示しています。画面が止まったら、ハングアップ成功です。リセットをかけてください。対象のトラックは30とします。

機械語

```
0000 B8711D     MOV AX,1D71H    ；単密度でフォーマット
0003 BB0010     MOV BX,1000H    ；4096バイト
0006 B91E05     MOV CX,051EH    ；セクタ長4096バイト、トラック30
0009 BAFC00     MOV DX,00FCH    ；サーフェス0、初期データFCH
000C 0E         PUSH CS
000D 07         POP ES
000E BD0001     MOV BP,100H
```

```
0011 CD1B       INT 1BH           ；BIOSコール
0013 B87176     MOV AX,7671H      ；倍密度でリードデータ、
                                  ；エラーリトライなし
0016 BB0001     MOV BX,100H
0019 B91E01     MOV CX,011EH
001C BA0100     MOV DX,0001H
001F CD1B       INT 1BH
0021 CF         IRET
```

```
BASIC
10 FOR I=1 TO 1000
20 PRINT"TRYING";I;"TIMES"
30   A=0：CALL A
40 NEXT
```

◎ チェック方法

ハングアップは、コピーしようとしたときに起こればよいのであって、実行時にはハングアップしては困ります。このプロテクトは、コピーができないようにするもので、実行できないというものではありません。ですから、実行時には、ハングアップするトラックへはアクセスしないようにしておけば、特にチェックルーチンは必要ありません。

◎ コピー方法

このプロテクトのかかったディスケットをハングアップせずに、コピーしようというのは、ほとんど不可能といえます。

考えられる方法は、ハングアップしたトラック番号を覚えておいて、リセット後リトライし、今度はそのトラックをとばしてコピーする、ということでしょう。

6.2 ハングアップの応用 ハングアップするかをチェック

◎ 原理

前節のプロテクトが、コピーツールをハングアップさせて、コピーを中断させることが目的であったのに対し、ここでは、さらにハングアップすることを実行時にチェックするプロテクトを考えます。

前節のコピー方法のところで述べた方法でコピーしたものは、ハングアップしたトラックを無視したことになりますが、ハングアップすることをチェックしてやれば、このコピーをもコピーとして判別することができます。

◎ 作成方法

前節とまったく同様にして作成してください。ただし、ハングアップするトラックのトラック番号を覚えておく必要があります。

◎ チェック方法

前節では、とくにチェックをしませんでしたが、ここでは、ハングアップするように処理したトラックを倍密度で読んで、ハングアップすることを確認します。ハングアップしたままではプログラムを実行できませんから、ハングアップを確認したらリセットして、プログラムを継続し、ハングアップしなければコピーであると判断して、プログラムの実行を中断します。

◎ コピー方法

前節の方法でコピーしますが、いちどハングアップしたトラックを無視するのではなく、読込みはしませんが、なんらかの方法でコピーツール側でもそのトラックをハングアップするように書き込みます。前節の作成方法と同じようにすればいいでしょう。

また、ハードウェアを用いたコピーツールでトラックごとコピーする場合も、ハングアップの条件を完全に維持するのは困難ですから、必ずしも実行できるコピーになるとはいいきれません。



PC-9800 SERIES THE PROTECT

III 付 録

○拡張BIOSの製作
○トラック容量測定と
調整ツール
○周辺LSIの使い方

1　付　録
拡張BIOSの製作

ＰＣ－９８００シリーズではＢＩＯＳが用意されていて、比較的細かなディスク入出力が、機械語で容易にできるようになっています。しかし、プロテクトに使うには少し不満が出てきます。

① フォーマットのときにＳＣ（セクタ数）や、ＧＰＬ（ＧＡＰ３の長さ）を指定できない。
② リセットをかけることができない。

そこで、これらの不満を解消すべくＢＩＯＳの機能を拡張することになりました。拡張ＢＩＯＳの制作にあたっては次のことを満たすよう考慮しました。

① ＰＣ－９８００シリーズのＢＩＯＳの持っている機能をすべて含むこと。つまり、完全な上位コンパチブルとする。
② コーリングシーケンス（パラメータの渡し方）もＢＩＯＳとコンパチブルとする。
③ 新しい機能のためのパラメータの引き渡しには、もともと使われていなかったレジスタ（ＳＩ，ＤＩ）を使用する。
④ 上にあげた不満を解消すること。

ここで紹介する拡張ＢＩＯＳは以上のような条件を満たすもので、読者の皆さんにもきっと満足していただけるものと思います。

使用可能なコマンド

ＢＩＯＳに含まれるコマンド（知識編第３章）のすべてです。そのうち機能が拡張されたのはライト系のコマンド（フォーマット、ライトデータ、ライトディリーテイッドデータの３つ）です。

コーリングシーケンス

ライト系を除くすべてのコマンドはＢＩＯＳとまったく同一です。今までのＢＩＯＳと思って使用してかまいません。ライト系コマンドでもＢＩＯＳの持っていたレジスタの意味はまったく同じです。従来のＢＩＯＳの使用法は知識編第３章を御覧ください。それに加えて、新しい機能のため次の２つのレジスタが追加されました。

ＳＩ

ＳＣ（またはＥＯＴ）とＧＰＬを指定します。

b15 … b8	b7 … b0
SC (or EOT)	GPL

上位８ビットでＳＣ（フォーマット時）またはＥＯＴ（ライトデータ時）を下位８ビットでＧＰＬを指定します。

フォーマット時にはＳＣは作られるセクタ数をＧＰＬは作られるＧＡＰ３の長さを意味します。

ライドデータ、ライトディリーテイッドデータ時には、ＥＯＴはマルチセクタ機能で処理される最終のセクタ番号を、ＧＰＬはデータライト後、読み飛ばされるバイト数を意味します。

ＤＩ

リセットをかけるタイミングを指定します。

ＤＭＡによるＦＤＣへのデータ転送が終了した時点を基準として、そこからリセットをかけるまでの時間です。ＤＩの１単位は約１バイトが書き込まれる時間に相当します。ＤＭＡが転送するバイト数は、ＢＸレジスタで指定できますから、ＢＸとＤＩの組み合わせで、ほとんど任意の点でリセットをかけることができます。リセット機能が有効となるのは、ライト系コマンドのみです。これらのコマンドでリセットをかけたくないときは、ＤＩを０ＦＦＦＦＨにしてください。

（例）ライトデータで、256バイトのセクタに50バイト書いたところでリセットをかけたい。

ＢＸ←50、　ＤＩ←３

とします。ここでＤＩを０にしないのは、データ転送を終わってからディスクに書き込まれるまでに３バイト分ぐらいの時間がかかるためです。つまり、50バイトを転送した時点ではＦＤＣは３バイト前の47バイト目を書き込んでいるということです。ですから、もし、ＤＩを０にすると、約47バイト書き込んだときにリセットがかかってしまいます。

（例）ライトデータで、512バイトのセクタに300バイト書いたところでリセットをかけたい。

ＢＸ←300　ＤＩ←３

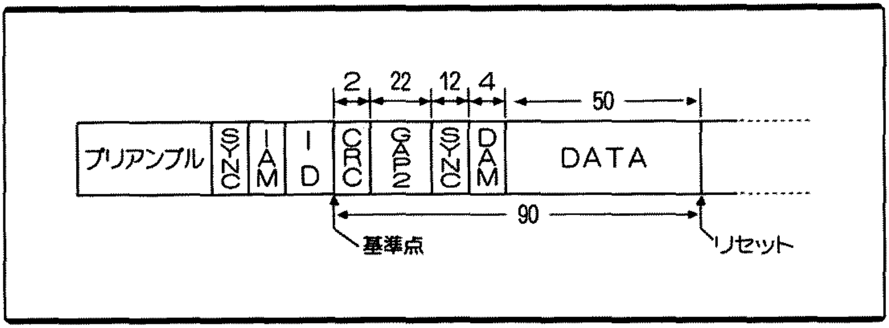
とします。

（例）フォーマットで、１番目のセクタ（256バイト）のデータを50バイト書いたところでリセットをかけたい。

ＢＸ←４　ＤＩ←93

とします。

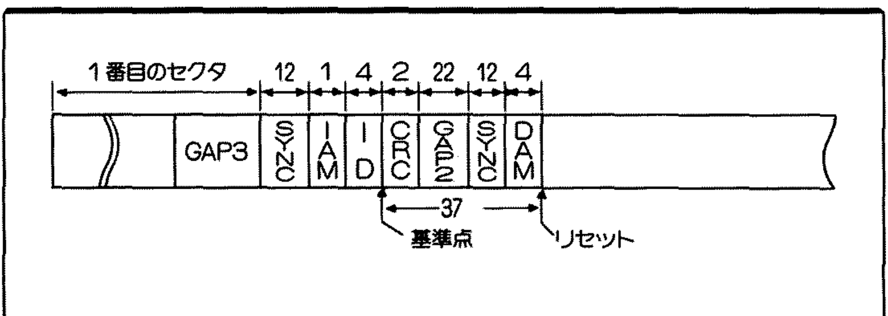
図1．1　リセットのタイミング



フォーマットのときにBXで指定するのはIDに書き込むデータの長さです。これは1セクタにつき4バイトですから、1番目のセクタでは、4となります。4バイトのデータ転送が終わった時点とはIDを書き込んだ時点（厳密には3バイトのずれ）で、そこが基準となります。リセットをかけたい位置は図1．1に示すように、そこから90バイト目ですから、遅れの3バイトを加えてDIは93にすればいいことがわかります。

例　フォーマットで2番目のセクタのDAMを書き込み中にリセットをかけたい。

図1．2　リセットのタイミング



BX←8、　DI←40

とします。

⊙ 呼び出し方法

ＢＩＯＳのＩＮＴ　１ＢＨの代わりにＩＮＴ　４０Ｈを使います。これは、この割り込みベクトルが通常使われていないため、４０Ｈを使っただけですから、もし、あなたがこのベクトルを他の目的に使用するのなら、変えてもかまいません。そのときは、ソースリストの３ページ目、上から15行目の４０Ｈを他の空いているベクトルに変えてください。

⊙ リターンコード

ＡＨにリターンコードが返ります。リセットがうまくかかったかどうかを知るため１つ追加してあります。

・リセットがかかった場合・・・・・・０１

・リセットがかからなかった場合・・・ＢＩＯＳと同じ

（知識編３．４参照）

⊙ 注　意

リセットがかかった場合、リキャリブレイトが行なわれるため、ヘッドは０トラックに移動します。続けてコマンドを与える場合はシークが必要です。御注意ください。

⊙ 準　備

拡張ＢＩＯＳを使うにあたっては、ＩＮＴ　４０Ｈのベクトルテーブルをあらかじめ拡張ＢＩＯＳのスタートアドレス（ラベル START）にセットしておく必要があります。セットをするためのプログラムも拡張ＢＩＯＳの先頭部分（ラベル INITIALIZE）に用意してありますので、それを使うと便利です。

N88－BASIC上で拡張BIOSを使うには次のようにしてください。

```
CLEAR ,&H××××
DEF SEG=&H××××
BLOAD "EBIOS.BIN",R
```

これだけで、拡張BIOSのロードとベクトルテーブルのセットが終了しますので、あとはBIOSと同じように自由に使えます。BIOSと拡張BIOSは同時に使えますから、目的に応じて使い分けてください。

⊙ プログラムのしくみ

このプログラムの特徴であるリセットをかける方法について解説します。リセットをかけること自体はIOアドレス（2DDの場合0CCH、2HDまたは8インチの場合94H）に88Hを書き込むだけです。問題はいかに正確にタイミングをとるかで、本プログラムではBIOSと同じようにDMAでデータ転送を行ない、DMACを監視することによってDMAのデータ転送の終了のタイミングを得て、そこを基準としてインターバルタイマを起動します。インターバルは1バイトの書き込み時間に合わせておきます。DIレジスタで指定した時間を経過したのち、リセットをかけるわけです。さらに興味のある方はソースリストをお読みください。

⊙ ソースリスト

ソースリストはMS－DOSのマクロアセンブラで記述してあります。直接、16進の機械語を入力したい方は、ダンプリストを使ってください。

```
                          ;*******************************
                          ;                              *
                          ;     EXTENDED BIOS PROGRAM    *
                          ;                              *
                          ;*******************************
                          ;
                          RETF      MACRO NUM
                                    LOCAL FARRET
                          FARRET    PROC  FAR
                                    RET   NUM
                          FARRET    ENDP
                                    ENDM
                          ;
                          DEVS     RECORD DEV:4,?01:1,HD:1,US:2
                          CMDS     RECORD MT:1,MFM:1,RT:1,SK:1,CMD:4
                          FLAG     RECORD CY:1
                          ;
                          REGS      STRUC
0000   ??                 _AL       DB    ?
0001   ??                 _AH       DB    ?
0002   ????               _BX       DW    ?
0004   ??                 _CL       DB    ?
0005   ??                 _CH       DB    ?
0006   ??                 _DL       DB    ?
0007   ??                 _DH       DB    ?
0008   ????               _BP       DW    ?
000A   ????               _ES       DW    ?
000C   ????               _DI       DW    ?
000E   ??                 _SIL      DB    ?
000F   ??                 _SIH      DB    ?
0010   ????               _DS       DW    ?
0012   ????????           _RETAD    DD    ?
0016   ??                 _PSWL     DB    ?
0017                      REGS      ENDS
                          ;
=                         CMND      EQU   _AH
=                         UNIT      EQU   _AL
=                         C         EQU   _CL
=                         H         EQU   _DH
=                         R         EQU   _DL
=                         N         EQU   _CH
=                         EOT       EQU   _SIH
=                         GPL       EQU   _SIL
=                         SC        EQU   _SIH
=                         D         EQU   _DL
=                         TIMER     EQU   _DI
=                         BUF_SEG   EQU   _ES
=                         BUF_OFF   EQU   _BP
=                         BUF_LEN   EQU   _BX
                          ;
                          FRDST    RECORD RQM:1,DIO:1,NDM:1,CB:1,BSYS:4
                          FDC_WC   RECORD RST:1,?2:2,DMAE:1,MTON:1,TMSK:1
                          ,?3:1,TTRG:1
```

```
                        FDC_RC RECORD RDY:1,?4:4
                        ;
                        FDC     STRUC
0000  ??                RD_ST   DB      ?
0001  ??                        DB      ?
0002  ??                WR_CMD  DB      ?
0003  ??                        DB      ?
0004  ??                WC      DB      ?
0005                    FDC     ENDS
                        ;
=                       RD_DT   EQU     WR_CMD
=                       RC      EQU     WC
                        ;
                        ;   IO ADDRESS DEFINITION
                        ;
= 0090                  FDC8    =       90H
= 00C8                  FDC5    =       0C8H
                        ;
= 0011                  DMAC_RS =       11H
= 0015                  MSK_REG =       15H
= 0017                  WRT_MOD =       17H
= 0019                  CLR_FF  =       19H
= 0009                  CH2_ADRS=       09H
= 000B                  CH2_CNT =       0BH
= 0023                  CH2_BANK=       23H
                        ;
= 0000                  PIC     =       0H
= 0002                  IMR     =       2H
                        ;
= 0077                  TMR_MOD =       77H
= 0071                  LOAD_CTR=       71H
                        ;
                        M?      RECORD M0:1
                        ;
                        ;    DEVICE CODE
                        ;
= 0070                  DEV_2DD =       70H
= 0090                  DEV_2HD =       90H
                        ;
                        ;     RETURN CODE
                        ;
= 0000                  NORMAL  =       0H
= 0001                  RST_DCT =       1H
= 0020                  BANK_OVER=      20H
= 0030                  END_CYL =       30H
= 0040                  EQ_CHECK=       40H
= 0050                  OVER_RUN=       50H
= 0060                  NOT_READY=      60H
= 0070                  NOT_WRT =       70H
= 00A0                  DATA_ERR=       0A0H
= 00C0                  NO_DATA =       0C0H
= 00E0                  MIS_ADM =       0E0H
                        ;
```

```
                          ST0  RECORD IC:2,SE:1,EC:1,NR:1,?HUS:3
                          ST1  RECORD EN:1,??:1,DE:1,OV:1,???:1,ND:1,
                          NW:1,MA:1
                          ;
                          ;     COMMAND CODE
                          ;
= 0003                    INIT      =       03H
= 0004                    SENCE_DEVICE=     04H
                          ;
= 00FF                    DTL       =       0FFH
                          ;
0000                      ZEROSEG SEGMENT AT 00000H
0020                              ORG 8*4
0020   ????????           INT8H   DD      ?
0100                              ORG 40H*4
0100   ????               INTOFF  DW      ?
0102   ????               INTSEG  DW      ?
                          ;
                          CLKFLG  RECORD CLK8:1,?4:7
                          ;
0501                              ORG     501H
0501   ??                 CLK_FLG DB      ?
055E                              ORG     55EH
055E   ??                 DSK_INT DB      ?
05CB                              ORG     5CBH
05CB   ??                 CNT_VAL DB      ?
05D0                              ORG     5D0H
05D0      07 (            RSLT    DB      7 DUP (?)
         ??
              )

05D7   ??                 MTR_CNT DB      ?
                          ;
05D8                      ZEROSEG ENDS
                          ;
0000                      CODE    SEGMENT BYTE PUBLIC
                                  ASSUME CS:CODE,DS:ZEROSEG
                          ;
0000                      INITIALIZE:
0000   33 C0                      XOR     AX,AX
0002   8E D8                      MOV     DS,AX
0004   C7 06 0100 R 0011 R        MOV     INTOFF,OFFSET START
000A   8C 0E 0102 R               MOV     INTSEG,CS
000E   CF                         IRET
                          ;
                          ;      WORK AREA
                          ;
000F   ????               MEDIA_FLG DW    ?
                          ;
0011                      START:
0011   FB                         STI
0012   FC                         CLD
0013   1E                         PUSH    DS
```

```
0014   56                      PUSH   SI
0015   57                      PUSH   DI
0016   06                      PUSH   ES
0017   55                      PUSH   BP
0018   52                      PUSH   DX
0019   51                      PUSH   CX
001A   53                      PUSH   BX
001B   50                      PUSH   AX
001C   8B EC                   MOV    BP,SP
001E   33 DB                   XOR    BX,BX
0020   8E DB                   MOV    DS,BX
                       ;
0022   33 FF                   XOR    DI,DI
0024   BA 0090                 MOV    DX,FDC8
0027   24 F0                   AND    AL,MASK DEV
0029   3C 90                   CMP    AL,DEV_2HD
002B   74 0B                   JE     L0
002D   3C 70                   CMP    AL,DEV_2DD
002F   75 32                   JNE    RETN
0031   BA 00C8                 MOV    DX,FDC5
0034   47                      INC    DI
0035   E8 009B R               CALL   MOTOR_ON
                       ;
                       ;    SEARCH COMMAND TABLE
                       ;     IF NOT FOUND CALL BIOS
                       ;
0038                   L0:
0038   2E: 89 3E 000F R        MOV    MEDIA_FLG,DI
003D   8A DC                   MOV    BL,AH
003F   81 E3 000F              AND    BX,MASK CMD
0043   D0 E3                   SHL    BL,1
0045   2E: 8B 9F 00C3 R        MOV    BX,WORD PTR CODE:CMD_TBL(BX)
004A   85 DB                   TEST   BX,BX
004C   74 3F                   JZ     BIOS_CALL
                       ;
                       ;    SEEK BIT CHECK
                       ;
004E   8A 46 01                MOV    AL,(BP).CMND
0051   A8 10                   TEST   AL,MASK SK
0053   74 0C                   JZ     NO_SEEK
                       ;
                       ;     BIOS SEEK CALL
                       ;
0055   8A 4E 04                MOV    CL,(BP).C
0058   8A 46 00                MOV    AL,(BP).UNIT
005B   B4 10                   MOV    AH,10H
005D   CD 1B                   INT    1BH
005F   72 02                   JB     RETN
0061                   NO_SEEK:
                       ;
                       ;    EXECUTE EACH COMMAND
                       ;
0061   FF D3                   CALL   BX
```

```
                               ;
                               ;       RETURN TO MAIN
                               ;
0063                           RETN:
0063  8B E5                             MOV    SP,BP
0065  A0 05CB R                         MOV    AL,CNT_VAL
0068  A2 05D7 R                         MOV    MTR_CNT,AL
006B  B0 09                             MOV    AL,MASK MTON OR MASK TTRG
006D  E6 CC                             OUT    FDC5.WC,AL
006F  B0 0C                             MOV    AL,MASK MTON OR MASK TMSK
0071  E6 CC                             OUT    FDC5.WC,AL
0073  80 66 16 FE                       AND    (BP)._PSWL,NOT MASK CY
0077  80 FC 20                          CMP    AH,20H
007A  72 04                             JB     NO_ERR
007C  80 4E 16 01                       OR     (BP)._PSWL,MASK CY
0080                           NO_ERR:
0080  88 66 01                          MOV    (BP)._AH,AH
0083  58                                POP    AX
0084  5B                                POP    BX
0085  59                                POP    CX
0086  5A                                POP    DX
0087  5D                                POP    BP
0088  07                                POP    ES
0089  5F                                POP    DI
008A  5E                                POP    SI
008B  1F                                POP    DS
008C  CF                                IRET
                               ;
                               ;        BIOS CALL
                               ;
008D                           BIOS_CALL:
008D  58                                POP    AX
008E  5B                                POP    BX
008F  59                                POP    CX
0090  5A                                POP    DX
0091  5D                                POP    BP
0092  07                                POP    ES
0093  5F                                POP    DI
0094  5E                                POP    SI
0095  1F                                POP    DS
0096  CD 1B                             INT    1BH
                                        RETF   2
0098                         + ??0000   PROC   FAR
0098  CA 0002                +          RET    2
009B                         + ??0000   ENDP
                               ;
                               ;       MOTOR ON
                               ;
009B                           MOTOR_ON:
009B  32 C0                             XOR    AL,AL
009D  86 06 05D7 R                      XCHG   AL,(MTR_CNT)
00A1  84 C0                             TEST   AL,AL
00A3  B0 08                             MOV    AL,MASK MTON
```

```
00A5  E6 CC                    OUT   FDC5.WC,AL
00A7  75 19                    JNZ   ALRDY
00A9  33 C9                    XOR   CX,CX
00AB  E2 FE                    LOOP  $
00AD  E2 FE                    LOOP  $
00AF  E2 FE                    LOOP  $
00B1  E2 FE                    LOOP  $
00B3  E2 FE                    LOOP  $
00B5  F6 06 0501 R 80          TEST  CLK_FLG,MASK CLK8
00BA  74 06                    JZ    ALRDY
00BC  E2 FE                    LOOP  $
00BE  E2 FE                    LOOP  $
00C0  E2 FE                    LOOP  $
00C2                   ALRDY:
00C2  C3                       RET
                       ;
                       ;    COMMAND TABLE
                       ;
00C3                   CMD_TBL:
00C3  0000             C_0     DW    0
00C5  0000             C_1     DW    0
00C7  0000             C_2     DW    0
00C9  0000             C_3     DW    0
00CB  0000             C_4     DW    0
00CD  0186 R           C_5     DW    WRITE_DATA
00CF  0000             C_6     DW    0
00D1  0000             C_7     DW    0
00D3  0000             C_8     DW    0
00D5  0186 R           C_9     DW    WRITE_DATA
00D7  0000             C_A     DW    0
00D9  0000             C_B     DW    0
00DB  0000             C_C     DW    0
00DD  01CA R           C_D     DW    WRITE_ID
00DF  0000             C_E     DW    0
00E1  0000             C_F     DW    0
                       ;
                       ;    COMMANDS SEND SUBROUTINE
                       ;
00E3                   DMAE_SEND_CMNDS:
00E3  B0 18                    MOV   AL,MASK DMAE OR MASK MTON
00E5  E6 CC                    OUT   FDC5.WC,AL
00E7  8A 66 01                 MOV   AH,(BP).CMND
00EA                   CMDS_SEND:
00EA  EC                       IN    AL,DX
00EB  A8 10                    TEST  AL,MASK CB
00ED  75 FB                    JNZ   CMDS_SEND
00EF  80 E4 CF                 AND   AH,MASK MT OR MASK MFM OR MASK
                       CMD
00F2  E8 0102 R                CALL  SEND
                       ;
                       ;   SEND 2ND BYTE
                       ;
00F5  8A 66 07                 MOV   AH,(BP).H
```

```
00F8  B1 02                 MOV   CL,HD
00FA  D2 E4                 SHL   AH,CL
00FC  32 66 00              XOR   AH,(BP).UNIT
00FF  80 E4 07              AND   AH,MASK HD OR MASK US
                    ;
                    ;     SEND TO FDC SUBROUTINE
                    ;
0102                SEND:
0102  EC                    IN    AL,DX
0103  34 80                 XOR   AL,MASK RQM
0105  A8 C0                 TEST  AL,MASK RQM OR MASK DIO
0107  75 F9                 JNZ   SEND
                    ;
0109  8A C4                 MOV   AL,AH
010B  52                    PUSH  DX
010C  83 C2 02              ADD   DX,WR_CMD
010F  EE                    OUT   DX,AL
0110  5A                    POP   DX
0111  C3                    RET
                    ;
                    ;     RECEIV RESULT STATUS
                    ;
0112                RCV_RESULT:
0112  EC                    IN    AL,DX
0113  A8 80                 TEST  AL,MASK RQM
0115  74 FB                 JZ    RCV_RESULT
0117  52                    PUSH  DX
0118  83 C2 02              ADD   DX,RD_DT
011B  EC                    IN    AL,DX
011C  5A                    POP   DX
011D  C3                    RET
                    ;
                    ;     DRIVE READY CHECK
                    ;
011E                RDY_CHECK:
011E  85 FF                 TEST  DI,DI
0120  74 10                 JZ    NONEED
0122  B4 04                 MOV   AH,SENCE_DEVICE
0124  E8 00EA R             CALL  CMDS_SEND
0127  E8 0112 R             CALL  RCV_RESULT
012A  E4 CC                 IN    AL,FDC5.RC
012C  A8 10                 TEST  AL,MASK RDY
012E  B4 60                 MOV   AH,NOT_READY
0130  74 01                 JZ    RETN_
0132  C3            NONEED: RET
0133  E9 0063 R     RETN_:  JMP   RETN
                    ;
                    ;     DMAC PARAMETER SET
                    ;
0136                SET_DMA:
0136  52                    PUSH  DX
0137  E4 11                 IN    AL,DMAC_RS
0139  8B C7                 MOV   AX,DI
```

```
013B  04 4A                ADD   AL,4AH
013D  E6 19                OUT   CLR_FF,AL
013F  90                   NOP
0140  E6 17                OUT   WRT_MOD,AL
0142  8B 46 0A             MOV   AX,(BP).BUF_SEG
0145  B1 04                MOV   CL,4
0147  D3 C0                ROL   AX,CL
0149  8A E8                MOV   CH,AL
014B  24 F0                AND   AL,0F0H
014D  03 46 08             ADD   AX,(BP).BUF_OFF
0150  80 D5 00             ADC   CH,0
0153  80 E5 0F             AND   CH,0FH
0156  50                   PUSH  AX
                        ;
                        ;    SET BASE ADDRESS
                        ;
0157  8B D7                MOV   DX,DI
0159  D1 E2                SHL   DX,1
015B  D1 E2                SHL   DX,1
015D  80 C2 09             ADD   DL,CH2_ADRS
0160  EE                   OUT   DX,AL
0161  8A C4                MOV   AL,AH
0163  EE                   OUT   DX,AL
                        ;
                        ;    SET COUNT (BX)-1
                        ;
0164  80 C2 02             ADD   DL,CH2_CNT-CH2_ADRS
0167  8B 46 02             MOV   AX,(BP).BUF_LEN
016A  48                   DEC   AX
016B  EE                   OUT   DX,AL
016C  86 C4                XCHG  AL,AH
016E  EE                   OUT   DX,AL
016F  86 C4                XCHG  AL,AH
                        ;
                        ;    CHECK BANK OVERFLOW
                        ;
0171  5B                   POP   BX
0172  03 C3                ADD   AX,BX
0174  9C                   PUSHF
                        ;
                        ;    SET  SEGMENT BANK
                        ;
0175  D1 EA                SHR   DX,1
0177  80 C2 1E             ADD   DL,CH2_BANK-CH2_CNT/2
017A  8A C5                MOV   AL,CH
017C  EE                   OUT   DX,AL
                        ;
                        ;    MASK OFF
                        ;
017D  8B C7                MOV   AX,DI
017F  04 02                ADD   AL,2
0181  E6 15                OUT   MSK_REG,AL
0183  9D                   POPF
```

```
0184  5A                   POP   DX
0185  C3                   RET
                      ;
                      ;    WRITE DATA COMMAND
                      ;
0186                  WRITE_DATA:
0186  E8 011E R            CALL  RDY_CHECK
0189  E8 0136 R            CALL  SET_DMA
018C  72 2F                JB    BNK_OVR
018E  E8 00E3 R            CALL  DMAE_SEND_CMNDS
0191  8A 66 04             MOV   AH,(BP).C
0194  E8 0102 R            CALL  SEND
0197  8A 66 07             MOV   AH,(BP).H
019A  E8 0102 R            CALL  SEND
019D  8A 66 06             MOV   AH,(BP).R
01A0  E8 0102 R            CALL  SEND
01A3  8A 66 05             MOV   AH,(BP).N
01A6  E8 0102 R            CALL  SEND
01A9  8A 66 0F             MOV   AH,(BP).EOT
01AC  E8 0102 R            CALL  SEND
01AF  8A 66 0E             MOV   AH,(BP).GPL
01B2  E8 0102 R            CALL  SEND
01B5  B4 FF                MOV   AH,DTL
01B7  E8 0102 R            CALL  SEND
01BA  EB 31 90             JMP   WAIT_DMA
                      ;
01BD                  BNK_OVR:
01BD  8B C7                MOV   AX,DI
01BF  04 06                ADD   AL,6
01C1  E6 15                OUT   MSK_REG,AL
01C3  B0 08                MOV   AL,MASK MTON
01C5  E6 CC                OUT   FDC5.WC,AL
01C7  B4 20                MOV   AH,BANK_OVER
01C9  C3                   RET
                      ;
                      ;    WRITE ID COMMAND
                      ;
01CA                  WRITE_ID:
01CA  E8 011E R            CALL  RDY_CHECK
01CD  E8 0136 R            CALL  SET_DMA
01D0  72 EB                JB    BNK_OVR
01D2  E8 00E3 R            CALL  DMAE_SEND_CMNDS
01D5  8A 66 05             MOV   AH,(BP).N
01D8  E8 0102 R            CALL  SEND
01DB  8A 66 0F             MOV   AH,(BP).SC
01DE  E8 0102 R            CALL  SEND
01E1  8A 66 0E             MOV   AH,(BP).GPL
01E4  E8 0102 R            CALL  SEND
01E7  8A 66 06             MOV   AH,(BP).D
01EA  E8 0102 R            CALL  SEND
                      ;
                      ;    WAIT DMA TC OR INT
                      ;
```

```
01ED                        WAIT_DMA:
01ED  8B CF                         MOV    CX,DI
01EF  B3 04                         MOV    BL,00000100B
01F1  D2 E3                         SHL    BL,CL
01F3  B9 0103                       MOV    CX,100H OR MASK US
01F6  22 4E 00                      AND    CL,(BP).UNIT
01F9  D2 E5                         SHL    CH,CL
01FB  8D B5 055E R                  LEA    SI,DSK_INT(DI)
01FF  85 FF                         TEST   DI,DI
0201  74 04                         JE     WAIT_TC
0203  B1 04                         MOV    CL,4
0205  D2 E5                         SHL    CH,CL
0207                        WAIT_TC:
0207  84 2C                         TEST   (SI),CH
0209  75 2E                         JNZ    INT_DCT
020B  E4 11                         IN     AL,DMAC_RS
020D  84 C3                         TEST   AL,BL
020F  74 F6                         JZ     WAIT_TC
0211  56                            PUSH   SI
0212  E8 02C9 R                     CALL   TIMER_SET
0215  5E                            POP    SI
0216                        WAIT_RST:
0216  84 2C                         TEST   (SI),CH
0218  75 12                         JNZ    INT_DCT0
021A  2E: F6 06 027B R FF           TEST   RST_FLG,0FFH
0220  74 F4                         JZ     WAIT_RST
0222  8A 46 00              RSTEND: MOV    AL,(BP).UNIT
0225  B4 03                         MOV    AH,INIT
0227  CD 1B                         INT    1BH
0229  B4 01                         MOV    AH,RST_DCT
022B  C3                            RET
022C                        INT_DCT0:
022C  57                            PUSH   DI
022D  E8 02B1 R                     CALL   RESTORE
0230  5F                            POP    DI
0231  2E: F6 06 027B R FF           TEST   RST_FLG,0FFH
0237  75 E9                         JNZ    RSTEND
0239                        INT_DCT:
0239  BE 05D0 R                     MOV    SI,OFFSET RSLT
023C  B4 00                         MOV    AH,NORMAL
023E  AC                            LODSB
023F  A8 C0                         TEST   AL,MASK IC
0241  74 2D                         JZ     N_ERR
0243  B4 40                         MOV    AH,EQ_CHECK
0245  A8 10                         TEST   AL,MASK EC
0247  75 27                         JNZ    N_ERR
0249  B4 60                         MOV    AH,NOT_READY
024B  A8 08                         TEST   AL,MASK NR
024D  75 21                         JNZ    N_ERR
024F  B4 30                         MOV    AH,END_CYL
0251  AC                            LODSB
0252  A8 80                         TEST   AL,MASK EN
0254  75 1A                         JNZ    N_ERR
```

```
0256  B4 A0                        MOV     AH,DATA_ERR
0258  A8 20                        TEST    AL,MASK DE
025A  75 14                        JNZ     N_ERR
025C  B4 50                        MOV     AH,OVER_RUN
025E  A8 10                        TEST    AL,MASK OV
0260  75 0E                        JNZ     N_ERR
0262  B4 C0                        MOV     AH,NO_DATA
0264  A8 04                        TEST    AL,MASK ND
0266  75 08                        JNZ     N_ERR
0268  B4 70                        MOV     AH,NOT_WRT
026A  A8 02                        TEST    AL,MASK NW
026C  75 02                        JNZ     N_ERR
026E  B4 E0                        MOV     AH,MIS_ADM
0270                     N_ERR:
0270  30 AD 055E R                 XOR     DSK_INT[DI],CH
0274  C3                           RET
                         ;
                         ;      RESET SUBROUTINE
                         ;
0275  ????               SAV_OFF  DW      ?
0277  ????               SAV_SEG  DW      ?
0279  ????               TMR_CNT  DW      ?
027B  ??                 RST_FLG  DB      ?
027C                     RESET:
027C  1E                           PUSH    DS
027D  57                           PUSH    DI
027E  52                           PUSH    DX
027F  50                           PUSH    AX
0280  33 C0                        XOR     AX,AX
0282  8E D8                        MOV     DS,AX
0284  2E: FF 0E 0279 R             DEC     TMR_CNT
0289  75 1B                        JNZ     NOT_ZERO
028B  2E: F6 06 000F R FF          TEST    BYTE PTR MEDIA_FLG,0FFH
0291  74 06                        JZ      F2HD
0293  B0 88                        MOV     AL,MASK RST OR MASK MTON
0295  E6 CC                        OUT     FDC5.WC,AL
0297  EB 04                        JMP     SHORT ?R1
0299  B0 D0              F2HD:     MOV     AL,0D0H
029B  E6 94                        OUT     FDC8.WC,AL
029D  E8 02B1 R          ?R1:      CALL    RESTORE
02A0  2E: C6 06 027B R FF          MOV     RST_FLG,0FFH
02A6                     NOT_ZERO:
02A6  FA                           CLI
02A7  B0 20                        MOV     AL,20H
02A9  E6 00                        OUT     PIC,AL
02AB  FB                           STI
02AC  58                           POP     AX
02AD  5A                           POP     DX
02AE  5F                           POP     DI
02AF  1F                           POP     DS
02B0  CF                           IRET
```

```
                              ;       MASK & RESTORE INT8H VECT
                              ;
02B1                          RESTORE:
02B1   E4 02                          IN      AL, IMR
02B3   0C 01                          OR      AL, MASK M0
02B5   E6 02                          OUT     IMR, AL
02B7   BF 0020 R                      MOV     DI, OFFSET INT8H
02BA   2E: A1 0275 R                  MOV     AX, SAV_OFF
02BE   89 05                          MOV     (DI), AX
02C0   47                             INC     DI
02C1   47                             INC     DI
02C2   2E: A1 0277 R                  MOV     AX, SAV_SEG
02C6   89 05                          MOV     (DI), AX
02C8   C3                             RET
                              ;
                              ;       TIMER SET ROUTINE
                              ;
02C9                          TIMER_SET:
02C9   BE 0020 R                      MOV     SI, OFFSET INT8H
02CC   B8 027C R                      MOV     AX, OFFSET RESET
02CF   FA                             CLI
02D0   87 04                          XCHG    (SI), AX
02D2   2E: A3 0275 R                  MOV     SAV_OFF, AX
02D6   46                             INC     SI
02D7   46                             INC     SI
02D8   8C C8                          MOV     AX, CS
02DA   87 04                          XCHG    (SI), AX
02DC   FB                             STI
02DD   2E: A3 0277 R                  MOV     SAV_SEG, AX
02E1   2E: C6 06 027B R 00            MOV     RST_FLG, 0
02E7   8B 46 0C                       MOV     AX, (BP).TIMER
02EA   2E: A3 0279 R                  MOV     TMR_CNT, AX
02EE   40                             INC     AX
02EF   74 1E                          JZ      RST_SKP
02F1   B0 34                          MOV     AL, 34H
02F3   E6 77                          OUT     TMR_MOD, AL
02F5   F6 06 0501 R 80                TEST    CLK_FLG, MASK CLK8
02FA   B8 0064                        MOV     AX, 100
02FD   75 02                          JNZ     CL_8M
02FF   B0 7B                          MOV     AL, 123
0301   E6 71                  CL_8M:  OUT     LOAD_CTR, AL
0303   8A C4                          MOV     AL, AH
0305   E6 71                          OUT     LOAD_CTR, AL
0307   FA                             CLI
0308   E4 02                          IN      AL, IMR
030A   24 FE                          AND     AL, NOT MASK M0
030C   E6 02                          OUT     IMR, AL
030E   FB                             STI
030F   C3                     RST_SKP:RET
0310                          CODE    ENDS
                                      END
```

ダンプリスト

このダンプリストの右端のＳＵＭは横一列を合計したものです。機械語入力エディタ（小社刊ＰＣ－９８００ユーティリティプログラム応用実例集に収録）を御利用になるとリアルタイムでＳＵＭチェックができます。

```
ADDR: +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F : SUM
0000: 33 C0 8E D8 C7 06 00 01 11 00 8C 0E 02 01 CF 00 : 4A4
0010: 00 FB FC 1E 56 57 06 55 52 51 53 50 8B EC 33 DB : 6E8
0020: 8E DB 33 FF BA 90 00 24 F0 3C 90 74 0B 3C 70 75 : 765
0030: 32 BA C8 00 47 E8 63 00 2E 89 3E 0F 00 8A DC 81 : 631
0040: E3 0F 00 D0 E3 2E 8B 9F C3 00 85 DB 74 3F 8A 46 : 7A3
0050: 01 A8 10 74 0C 8A 4E 04 8A 46 00 B4 10 CD 1B 72 : 503
0060: 02 FF D3 8B E5 A0 CB 05 A2 D7 05 B0 09 E6 CC B0 : 94D
0070: 0C E6 CC 80 66 16 FE 80 FC 20 72 04 80 4E 16 01 : 6AF
0080: 88 66 01 58 5B 59 5A 5D 07 5F 5E 1F CF 58 5B 59 : 570
0090: 5A 5D 07 5F 5E 1F CD 1B CA 02 00 32 C0 86 06 D7 : 5A3
00A0: 05 84 C0 B0 08 E6 CC 75 19 33 C9 E2 FE E2 FE E2 : 9DF
00B0: FE E2 FE E2 FE F6 06 01 05 80 74 06 E2 FE E2 FE : A7A
00C0: E2 FE C3 00 00 00 00 00 00 00 00 00 00 86 01 00 : 32A
00D0: 00 00 00 00 00 86 01 00 00 00 00 00 00 CA 01 00 : 152
00E0: 00 00 00 B0 18 E6 CC 8A 66 01 EC A8 10 75 FB 80 : 6FF
00F0: E4 CF E8 0D 00 8A 66 07 B1 02 D2 E4 32 66 00 80 : 720
0100: E4 07 EC 34 80 A8 C0 75 F9 8A C4 52 83 C2 02 EE : 936
0110: 5A C3 EC A8 80 74 FB 52 83 C2 02 EC 5A C3 85 FF : 9C6
0120: 74 10 B4 04 E8 C3 FF E8 E8 FF E4 CC A8 10 B4 60 : A31
0130: 74 01 C3 E9 2D FF 52 E4 11 8B C7 04 4A E6 19 90 : 7C3
0140: E6 17 8B 46 0A B1 04 D3 C0 8A E8 24 F0 03 46 08 : 6F7
0150: 80 D5 00 80 E5 0F 50 8B D7 D1 E2 D1 E2 80 C2 09 : 92C
0160: EE 8A C4 EE 80 C2 02 8B 46 02 48 EE 86 C4 EE 86 : 935
0170: C4 5B 03 C3 9C D1 EA 80 C2 1E 8A C5 EE 8B C7 04 : 92F
0180: 02 E6 15 9D 5A C3 E8 95 FF E8 AA FF 72 2F E8 52 : 99F
0190: FF 8A 66 04 E8 6B FF 8A 66 07 E8 65 FF 8A 66 06 : 87E
01A0: E8 5F FF 8A 66 05 E8 59 FF 8A 66 0F E8 53 FF 8A : 93E
01B0: 66 0E E8 4D FF B4 FF E8 48 FF EB 31 90 8B C7 04 : 98C
01C0: 06 E6 15 B0 08 E6 CC B4 20 C3 E8 51 FF E8 66 FF : 987
01D0: 72 EB E8 0E FF 8A 66 05 E8 27 FF 8A 66 0F E8 21 : 85D
01E0: FF 8A 66 0E E8 1B FF 8A 66 06 E8 15 FF 8B CF B3 : 8FE
01F0: 04 D2 E3 B9 03 01 22 4E 00 D2 E5 8D B5 5E 05 85 : 6C7
0200: FF 74 04 B1 04 D2 E5 84 2C 75 2E E4 11 84 C3 74 : 7E6
0210: F6 56 E8 B4 00 5E 84 2C 75 12 2E F6 06 7B 02 FF : 723
0220: 74 F4 8A 46 00 B4 03 CD 1B B4 01 C3 57 E8 81 00 : 70F
0230: 5F 2E F6 06 7B 02 FF 75 E9 BE D0 05 B4 00 AC A8 : 7FE
0240: C0 74 2D B4 40 A8 10 75 27 B4 60 A8 08 75 21 B4 : 6B7
0250: 30 AC A8 80 75 1A B4 A0 A8 20 75 14 B4 50 A8 10 : 6F4
0260: 75 0E B4 C0 A8 04 75 08 B4 70 A8 02 75 02 B4 E0 : 6F9
0270: 30 AD 5E 05 C3 00 00 00 00 00 00 00 1E 57 52 50 : 31A
0280: 33 C0 8E D8 2E FF 0E 79 02 75 1B 2E F6 06 0F 00 : 5D8
```

```
0290: FF 74 06 B0 88 E6 CC EB 04 B0 D0 E6 94 E8 11 00 : 945
02A0: 2E C6 06 7B 02 FF FA B0 20 E6 00 FB 58 5A 5F 1F : 751
02B0: CF E4 02 0C 01 E6 02 BF 20 00 2E A1 75 02 89 05 : 55D
02C0: 47 47 2E A1 77 02 89 05 C3 BE 20 00 B8 7C 02 FA : 635
02D0: 87 04 2E A3 75 02 46 46 8C C8 87 04 FB 2E A3 77 : 681
02E0: 02 2E C6 06 7B 02 00 8B 46 0C 2E A3 79 02 40 74 : 456
02F0: 1E B0 34 E6 77 F6 06 01 05 80 B8 64 00 75 02 B0 : 624
0300: 7B E6 71 8A C4 E6 71 FA E4 02 24 FE E6 02 FB C3 : A1F
```

2 付　録

トラック容量測定と調整ツール

応用編第4章でみてきたように、トラック容量の差を使ったプロテクトでは、トラック容量を合わせない限りコピーはできないわけです。トラック容量を合わせるためには、まず、コピーする対象のトラック容量、つまり、そのトラックに何バイト書かれているか、を知る必要があります。次に同じトラック容量となるように自分のドライブ装置の回転数を調整してフォーマットすれば、これらのプロテクトにも対処できます。ここで紹介するプログラムは、未知のディスケットの指定したトラックのトラック容量を測定するプログラムです。

⊙ 使用法

① 未知のディスケットのトラック容量を測定します。プログラムAを使います。未知のディスケットをドライブに入れ、測定したいトラックを指定します。ただちにトラック容量を測定して表示します。

② ①で測定したトラック容量を覚えておきます。同じトラック容量となるようにドライブ装置の回転数を調整します。プログラムBを使います。プログラムBはフォーマットと測定を交互に繰り返すプログラムで、自分のドライブ装置で作成されるトラック容量をリアルタイムで知ることができます。もし、回転数を調整できるなら、先に覚えておいたトラック容量と一致するように調整します。後は普通にコピーすればいいでしょう。

もし、回転数を調整できないなら、自分のドライブ装置とどの程度差があるか計ってください。20バイト以内なら偶然の変動でうまくいくことがあります。

ドライブ装置が異なればかなりトラック容量も異なりますから、何台もある人は試してください。それでもだめならそのプロテクトは解けません。あきらめてください。

⦿——測定の原理

このプログラムでは、おおまかな測定と精密な測定の2段階で行ないます。

まず、時間測定によっておおまかな測定を行ないます。

① あなたのドライブ装置の1回転に要する時間を測定します。これをaとします。

② リードダイアグでxバイト（多ければ何バイトでもよい）読み出す時間を測定します。これをbとします。

以上より　トラック容量：a＝x：b
となり、トラック容量は次式で求まります。

トラック容量＝x＊a／b

次に精密な測定をパターンマッチングによって行ないます。リードダイアグで1周以上のデータを読み込み先頭セクタ部分のパターンを見つけ、パターンの見つかった位置から1周のバイト数が求まります。

先頭セクタのＳＹＮＣとＤＡＭの部分でパターンマッチングを行ないます。ただし、1周読み込みますから途中でビットずれを起こしているはずで、あらゆるビットずれを想定しなくてはなりません。ＳＹＮＣ＋ＤＡＭの部分はビットずれで次のように変化します。

表2．1　SYNC＋DAMのビットずれパターン

ビットずれ	パターン（16進数）	最終バイトの有効ビット数
0.0	00 00 00……00 00 00 A1 A1 A1 FB	8
0.5	FF FF FF……FF FF FF 14 14 14 00	8
1.0	00 00 00……00 00 01 43 43 43 F6	7
1.5	FF FF FF……FF FF FC 28 28 28 00	7
2.0	00 00 00……00 00 02 86 86 87 EC	6
2.5	FF FF FF……FF FF F8 50 50 50 00	6
3.0	00 00 00……00 00 05 0D 0D 0F D8	5
3.5	FF FF FF……FF FF F0 A0 A0 A0 00	5
4.0	00 00 00……00 00 0A 1A 1A 1F B	4
4.5	FF FF FF……FF FF E1 41 41 40 0	4
5.0	00 00 00……00 00 14 34 34 3F 6	3
5.5	FF FF FF……FF FF C2 82 82 80 0	3
6.0	00 00 00……00 00 28 68 68 7E C	2
6.5	FF FF FF……FF FF 85 05 05 00 0	2
7.0	00 00 00……00 00 50 D0 D0 FD 8	1
7.5	FF FF FF……FF FF 0A 0A 0A 00 0	1

また、SYNC＋DAMのパターンは各セクタの先頭にありますから、時間測定から求めたトラック容量のあたりで、これらのパターンのいずれかとマッチングがとれたとき、DAMの最終バイトまでのバイト数をトラック容量とします。

⊙ソースリスト

機械語（プログラムA，B共通）

MS－DOSのマクロアセンブラで記述してあります。そのまま使用されるときはダンプリストを御使用ください。

```
                                ;*****************************
                                ;                            *
                                ;   KAITEN SOKUTEI PROGRAM   *
                                ;                            *
                                ;*****************************
                                ;
= 0032                          HANI    =       50
                                ;
0000                            ZEROSEG SEGMENT AT 00000H
0501                                    ORG     501H
0501  ??                        CLK_FLG DB      ?
0502                            ZEROSEG ENDS
                                ;
0000                            CODE    SEGMENT BYTE PUBLIC
                                        ASSUME CS:CODE
0000  E8 00E4 R                 START:  CALL    TIMER_SET
0003  0E                                PUSH    CS
0004  1F                                POP     DS
                                        ASSUME DS:CODE
0005  B4 72                             MOV     AH,72H
0007  A0 0149 R                         MOV     AL,DRIVE
000A  BB 0100                           MOV     BX,100H
000D  B5 01                             MOV     CH,1
000F  8A 0E 014A R                      MOV     CL,TRACK
0013  8A 36 014B R                      MOV     DH,SURFACE
0017  0E                                PUSH    CS
0018  07                                POP     ES
0019  BD 0200 R                         MOV     BP,OFFSET BUFF
001C  CD 1B                             INT     1BH
001E  C7 06 014C R 0000                 MOV     TIME,0
0024  72 52                             JB      ERROR1
0026  B4 72                             MOV     AH,72H
0028  CD 1B                             INT     1BH
002A  8B 1E 014C R                      MOV     BX,TIME
002E  8B FB                             MOV     DI,BX
0030  B4 72                             MOV     AH,72H
0032  BB 2000                           MOV     BX,2000H
0035  B5 06                             MOV     CH,6
0037  A8 80                             TEST    AL,80H
0039  74 04                             JZ      D2DD
003B  D1 E3                             SHL     BX,1
003D  FE C5                             INC     CH
003F  CD 1B                     D2DD:   INT     1BH
0041  8B 0E 014C R                      MOV     CX,TIME
0045  8B C7                             MOV     AX,DI
0047  2B C8                             SUB     CX,AX
0049  2B C8                             SUB     CX,AX
004B  81 EB 0100                        SUB     BX,100H
004F  F7 E3                             MUL     BX
0051  F7 F1                             DIV     CX
0053  8B F0                             MOV     SI,AX
0055  81 C6 0200 R                      ADD     SI,OFFSET BUFF
0059  E8 012B R                         CALL    RESTORE
```

```
005C  8D 7C 32                    LEA     DI,(SI+HANI)
005F  83 EE 32                    SUB     SI,HANI
                         ;
                         ;   PRECISE MEASUREMENT
                         ;
0062  B3 FF              LOOP:    MOV     BL,0FFH
0064  38 1C              L0:      CMP     (SI),BL
0066  74 13                       JE      L1
0068  FE C3                       INC     BL
006A  74 F8                       JZ      L0
006C  46                          INC     SI
006D  3B F7                       CMP     SI,DI
006F  72 F1                       JB      LOOP
0071  C7 06 014E R 0000           MOV     BYTES,0
0077  CF                          IRET
0078  EB 5C 90           ERROR1:  JMP     ERROR
                         ;
                         ;  SYNC LENGTH CHECK
                         ;
007B  8B D6              L1:      MOV     DX,SI
007D  46                 L2:      INC     SI
007E  38 1C                       CMP     (SI),BL
0080  74 FB                       JE      L2
0082  8B C6                       MOV     AX,SI
0084  2B C2                       SUB     AX,DX
0086  3D 000A                     CMP     AX,10
0089  72 D7                       JB      LOOP
008B  3D 000F                     CMP     AX,15
008E  73 D2                       JNB     LOOP
                         ;
                         ;   PATTERN MATCHING
                         ;
0090  B8 00A1                     MOV     AX,00A1H
0093  BA A1F8                     MOV     DX,0A1F8H
0096  80 FB 00                    CMP     BL,0
0099  74 06                       JE      L3
009B  B8 FF0A                     MOV     AX,0FF0AH
009E  BA 0A00                     MOV     DX,0A00H
00A1  B9 0008            L3:      MOV     CX,8
00A4  BB FFFC                     MOV     BX,0FFFCH
00A7  D1 E2              L4:      SHL     DX,1
00A9  D1 D0                       RCL     AX,1
00AB  D1 E3                       SHL     BX,1
00AD  38 24                       CMP     (SI),AH
00AF  74 04                       JE      L5
00B1  E2 F4                       LOOP    L4
00B3  EB AD                       JMP     LOOP
00B5  46                 L5:      INC     SI
00B6  38 04                       CMP     (SI),AL
00B8  75 A8                       JNE     LOOP
00BA  46                          INC     SI
00BB  38 04                       CMP     (SI),AL
00BD  75 A3                       JNE     LOOP
```

```
00BF  46                                INC     SI
00C0  22 3C                             AND     BH, (SI)
00C2  3A F7                             CMP     DH, BH
00C4  75 9C                             JNE     LOOP
00C6  46                                INC     SI
00C7  22 1C                             AND     BL, (SI)
00C9  3A D3                             CMP     DL, BL
00CB  75 95                             JNE     LOOP
00CD  81 EE 01FF R                      SUB     SI,OFFSET BUFF-1
00D1  89 36 014E R                      MOV     BYTES,SI
00D5  CF                                IRET
                             ;
00D6  E8 012B R              ERROR:     CALL    RESTORE
00D9  C7 06 014E R 0001                 MOV     BYTES,1
00DF  CF                                IRET
                             ;
                             ;     TIMER SET ROUTINE
                             ;
00E0  ????                   SAV_OFF DW         ?
00E2  ????                   SAV_SEG DW         ?
                             ;
00E4                         TIMER_SET:
00E4  33 C0                             XOR     AX,AX
00E6  8E D8                             MOV     DS,AX
                                        ASSUME  DS:ZEROSEG
00E8  BE 0020                           MOV     SI,0020H
00EB  B8 011F R                         MOV     AX,OFFSET INTRPT
00EE  FA                                CLI
00EF  87 04                             XCHG    (SI),AX
00F1  2E: A3 00E0 R                     MOV     SAV_OFF,AX
00F5  46                                INC     SI
00F6  46                                INC     SI
00F7  8C C8                             MOV     AX,CS
00F9  87 04                             XCHG    (SI),AX
00FB  FB                                STI
00FC  2E: A3 00E2 R                     MOV     SAV_SEG,AX
0100  B0 34                             MOV     AL,34H
0102  E6 77                             OUT     77H,AL
0104  F6 06 0501 R 80                   TEST    CLK_FLG,80H
0109  B8 0064                           MOV     AX,100
010C  75 02                             JNZ     CL_8M
010E  B0 7B                             MOV     AL,123
0110  E6 71                  CL_8M:     OUT     71H,AL
0112  8A C4                             MOV     AL,AH
0114  E6 71                             OUT     71H,AL
0116  FA                                CLI
0117  E4 02                             IN      AL,02
0119  24 FE                             AND     AL,0FEH
011B  E6 02                             OUT     02,AL
011D  FB                                STI
011E  C3                                RET
                             ;
                             ;       INTERRUPT ROUTINE
```

```
011F                            INTRPT:
011F  2E: FF 06 014C R                  INC     CS:TIME
0124  50                                PUSH    AX
0125  B0 20                             MOV     AL,20H
0127  E6 00                             OUT     00,AL
0129  58                                POP     AX
012A  CF                                IRET
                                ;
                                ;    RESTORE ROUTINE
                                ;
012B                            RESTORE:
012B  1E                                PUSH    DS
012C  33 C0                             XOR     AX,AX
012E  8E D8                             MOV     DS,AX
                                        ASSUME  DS:ZEROSEG
0130  E4 02                             IN      AL,02
0132  0C 01                             OR      AL,01
0134  E6 02                             OUT     02,AL
0136  BF 0020                           MOV     DI,20H
0139  2E: A1 00E0 R                     MOV     AX,SAV_OFF
013D  89 05                             MOV     [DI],AX
013F  47                                INC     DI
0140  47                                INC     DI
0141  2E: A1 00E2 R                     MOV     AX,SAV_SEG
0145  89 05                             MOV     [DI],AX
0147  1F                                POP     DS
0148  C3                                RET
                                ;
                                ;   WORK AREA
                                ;
0149  ??                        DRIVE   DB      ?
014A  ??                        TRACK   DB      ?
014B  ??                        SURFACE DB      ?
014C  ????                      TIME    DW      ?
014E  ????                      BYTES   DW      ?
                                ;
                                ;    FORMAT ROUTINE
                                ;
0150  B4 5D                             MOV     AH,5DH
0152  2E: A0 0149 R                     MOV     AL,DRIVE
0156  BB 0004                           MOV     BX,4
0159  B5 01                             MOV     CH,01
015B  2E: 8A 0E 014A R                  MOV     CL,TRACK
0160  2E: 8A 36 014B R                  MOV     DH,SURFACE
0165  0E                                PUSH    CS
0166  07                                POP     ES
0167  BD 0000                           MOV     BP,0
016A  CD 1B                             INT     1BH
016C  CF                                IRET
                                ;
0200                                    ORG     200H
0200  2000 [ ?? ]               BUFF    DB      2000H DUP (?)
2200                            CODE    ENDS
```

BASIC

```
10 'プログラム A
20 INPUT"5インチOR8インチ（2HD含む）(5/8)=";MDA
30 INPUT"ドライブ番号=";DRV
40 IF MDA=5 THEN DRV=DRV+&H70-1 ELSE DRV=DRV+&H90-1
50 POKE &H149,DRV
60 INPUT"トラック番号=";TRK : POKE &H14A,TRK
70 INPUT"サーフェス番号=";SUR:POKE &H14B,SUR
80 A=0 : CALL A
90 BYTES=PEEK(&H14E)+256*PEEK(&H14F)
100 IF BYTES=1 THEN PRINT"リードダイアグ失敗":END
110 IF BYTES=0 THEN PRINT"パターンマッチング失敗":GOTO 80
120 PRINT"トラック容量=";BYTES;"バイト"
```

```
10 'プログラム B
20 CLS : ON STOP GOSUB *STOP.KEY : STOP ON
30 INPUT"5インチOR8インチ（2HD含む）(5/8)=";MDA
40 INPUT"ドライブ番号=";DRV
50 IF MDA=5 THEN DRV=DRV+&H70-1 ELSE DRV=DRV+&H90-1
60 POKE &H149,DRV
70 INPUT"トラック番号=";TRK : POKE &H14A,TRK
80 INPUT"サーフェス番号=";SUR:POKE &H14B,SUR
90 B=&H150 : CALL B
100 A=0 : CALL A
110 BYTES=PEEK(&H14E)+256*PEEK(&H14F)
120 LOCATE 30,10
130 IF BYTES=1 THEN PRINT"リードダイアグ失敗":END
140 IF BYTES=0 THEN PRINT"パターンマッチング失敗":END
150 PRINT"トラック容量=";BYTES;"バイト"
160 GOTO 90
170 *STOP.KEY : STOP OFF :END
```

ダンプリスト

FILE NAME=KAITEN.COM

ADDR:	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F	:	SUM
0000:	E8	E1	00	0E	1F	B4	72	A0	49	01	BB	00	01	B5	01	8A	:	602
0010:	0E	4A	01	8A	36	4B	01	0E	07	BD	00	02	CD	1B	C7	06	:	3EE
0020:	4C	01	00	00	72	52	B4	72	CD	1B	8B	1E	4C	01	8B	FB	:	59B
0030:	B4	72	BB	00	20	B5	06	A8	80	74	04	D1	E3	FE	C5	CD	:	8A0
0040:	1B	8B	0E	4C	01	8B	C7	2B	C8	2B	C8	81	EB	00	01	F7	:	69D
0050:	E3	F7	F1	8B	F0	81	C6	00	02	E8	CF	00	8D	7C	32	83	:	904
0060:	EE	32	B3	FF	38	1C	74	13	FE	C3	74	F8	46	3B	F7	72	:	8C4
0070:	F1	C7	06	4E	01	00	00	CF	EB	5C	90	8B	D6	46	38	1C	:	6AE
0080:	74	FB	8B	C6	2B	C2	3D	0A	00	72	D7	3D	0F	00	73	D2	:	6CE
0090:	B8	A1	00	BA	F8	A1	80	FB	00	74	06	B8	0A	FF	BA	00	:	81C
00A0:	0A	B9	08	00	BB	FC	FF	D1	E2	D1	D0	D1	E3	38	24	74	:	959
00B0:	04	E2	F4	EB	AD	46	38	04	75	A8	46	38	04	75	A3	46	:	6F1
00C0:	22	3C	3A	F7	75	9C	46	22	1C	3A	D3	75	95	81	EE	FF	:	7A9
00D0:	01	89	36	4E	01	CF	E8	52	00	C7	06	4E	01	01	00	CF	:	504
00E0:	00	00	00	00	33	C0	8E	D8	BE	20	00	B8	1F	01	FA	87	:	590
00F0:	04	2E	A3	E0	00	46	46	8C	C8	87	04	FB	2E	A3	E2	00	:	6CE
0100:	B0	34	E6	77	F6	06	01	05	80	B8	64	00	75	02	B0	7B	:	681
0110:	E6	71	8A	C4	E6	71	FA	E4	02	24	FE	E6	02	FB	C3	2E	:	9D2
0120:	FF	06	4C	01	50	B0	20	E6	00	58	CF	1E	33	C0	8E	D8	:	6F6
0130:	E4	02	0C	01	E6	02	BF	20	00	2E	A1	E0	00	89	05	47	:	53E
0140:	47	2E	A1	E2	00	89	05	1F	C3	00	00	00	00	00	00	00	:	368
0150:	B4	5D	2E	A0	49	01	BB	04	00	B5	01	2E	8A	0E	4A	01	:	4AF
0160:	2E	8A	36	4B	01	0E	07	BD	00	00	CD	1B	CF	00	00	00	:	3C3

3 付　録
周辺LSIの使い方

プロテクトに関連して使うことがあると思われる周辺ＬＳＩについて解説します。ここではＦＤＣ，ＤＭＡ，割り込み、タイマをとりあげます。

◉ ＦＤＣ

μＰＤ７６５ＡというＬＳＩが使われています。また、ＦＤＣ以外に関連のＩＯポートとしては、モータのＯＮ／ＯＦＦ，リセットを操作するコントロールレジスタがあります。詳しい解説は知識編第２章で行ないましたので、ここでは、ＩＯアドレスとコマンド、パラメータの一覧を掲げます。

表３．１　ＩＯアドレス

ＩＯアドレス		Ｒ／Ｗ	内　容
２ＤＤ	２ＨＤ		
Ｃ８	９０	Ｒ	ステータスレジスタ・リード
ＣＡ	９２	Ｒ	データレジスタ・リード
ＣＡ	９２	Ｗ	データレジスタ・ライト
ＣＣ	９４	Ｗ	コントロールレジスタ・ライト

ステータスレジスタは、ＦＤＣにコマンド、パラメータを書き込むとき、バッファが空いているかを調べるのに使います。コマンド等はデータレジスタに書き込みます。コントロールレジスタのビット７（最上位ビット）はリセットで、ここに１を書き込むとリセットがかかります。

<table>
<tr><th>コマンド</th><th>D_7</th><th>D_6</th><th>D_5</th><th>D_4</th><th>D_3</th><th>D_2</th><th>D_1</th><th>D_0</th></tr>
<tr><td rowspan="9">Read
Diagnostic</td><td>MT</td><td>MF</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>HD</td><td>US1</td><td>US0</td></tr>
<tr><td colspan="8">←C→</td></tr>
<tr><td colspan="8">←H→</td></tr>
<tr><td colspan="8">←R→</td></tr>
<tr><td colspan="8">←N→</td></tr>
<tr><td colspan="8">←EOT→</td></tr>
<tr><td colspan="8">←GPL→</td></tr>
<tr><td colspan="8">←DTL→</td></tr>
<tr><td rowspan="3">Specify</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td colspan="4">←SRT→</td><td colspan="4">←HUT→</td></tr>
<tr><td colspan="7">←HLT→</td><td>ND</td></tr>
<tr><td rowspan="2">Sense
Device
Status</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>HD</td><td>US1</td><td>US0</td></tr>
<tr><td rowspan="9">Write
Data</td><td>MT</td><td>MF</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>HD</td><td>US1</td><td>US0</td></tr>
<tr><td colspan="8">←C→</td></tr>
<tr><td colspan="8">←H→</td></tr>
<tr><td colspan="8">←R→</td></tr>
<tr><td colspan="8">←N→</td></tr>
<tr><td colspan="8">←EOT→</td></tr>
<tr><td colspan="8">←GPL→</td></tr>
<tr><td colspan="8">←DTL→</td></tr>
</table>

コ マ ン ド	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Read Data	MT	MF	SK	0	0	1	1	0
	X	X	X	X	X	HD	US1	US0
	← C →							
	← H →							
	← R →							
	← N →							
	← EOT →							
	← GPL →							
	← DTL →							
Recalibrate	0	0	0	0	0	1	1	1
	X	X	X	X	X	0	US1	US0
Sence Int Status	0	0	0	0	1	0	0	0
Write Deleted Data	MT	MF	0	0	1	0	0	1
	X	X	X	X	X	HD	US1	US0
	← C →							
	← H →							
	← R →							
	← N →							
	← EOT →							
	← GPL →							
	← DTL →							

コマンド	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Read ID	0	MF	0	0	1	0	1	0
	X	X	X	X	X	HD	US1	US0
Read Deleted Data	MT	MF	SK	0	1	1	0	0
	X	X	X	X	X	HD	US1	US0
	←C→							
	←H→							
	←R→							
	←N→							
	←EOT→							
	←GPL→							
	←DTL→							
Write ID	0	MF	0	0	1	1	0	1
	X	X	X	X	X	HD	US1	US0
	←N→							
	←SC→							
	←GPL→							
	←D→							
Seek	0	0	0	0	1	1	1	1
	X	X	X	X	X	HD	US1	US0
	←NCN→							

以下はB1OSでは使わないコマンドです。

コマンド	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Scan Equal	MT	MF	SK	1	0	0	0	1
	X	X	X	X	X	HD	US1	US0
	←C→							
	←H→							
	←R→							
	←N→							
	←EOT→							
	←GPL→							
	←STP→							
Scan Low or Equal	MT	MF	SK	1	1	0	0	1
	X	X	X	X	X	HD	US1	US0
	←C→							
	←H→							
	←R→							
	←N→							
	←EOT→							
	←GPL→							
	←STP→							

（注） STPは比較すべきセクタのセクタ間隔で1または2をとります。1のときは連続、2のときは1つおきです。

コマンド	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Scan High	MT	MF	SK	1	1	1	0	1
or Equal	X	X	X	X	X	HD	US1	US0
	←C→							
	←H→							
	←R→							
	←N→							
	←EOT→							
	←GPL→							
	←STP→							

⊙ DMAC

DMAのコントロール用にμPD８２３７AというＬＳＩが使われています。ディスク入出力の際、ＦＤＣとメモリとのデータ転送はDMAで行なわれています。DMACには０～３の４チャンネルありますが、ディスク用にはチャンネル２が８インチまたは２ＨＤ用に、チャンネル３が２ＤＤ用に使われています。

DMACからのアドレスバスは16本しかなく、アドレス空間は64Ｋバイトです。PC－９８００シリーズでは、そこで上位４ビットを付け加え、全アドレス空間をカバーするようにしています。これをバンクと呼びます。しかし、バンクは転送中に変更できませんから、バッファ領域が64Ｋバイト境界をまたぐような転送はできないことになります。

DMACの制御は次の手順で行ないます。

① クリアバイトポインタフリップフロップ

② モードの設定

③ 転送開始アドレスの設定

上位4ビットはバンクに、下位16ビットは下位、上位の順でアドレスレジスタに設定する

④ 転送バイト数の設定

下位、上位バイトの順でカウントレジスタに設定する

⑤ マスクのクリア

⊙ 割り込み

割り込みのコントロール用にはμＰＤ８２５９ＡというＬＳＩが２つ使われています。プロテクト関連では、ＦＤＣの動作終了を割り込みによってＣＰＵに知らせる、インターバルタイマで一定時間ごとに割り込みをかけるなどに使われます。ＦＤＣ、タイマなどから出された割り込み要求はすべてこの割り込みコントローラに入り、割り込みコントローラでは優先順位をつけてＣＰＵに割り込み要求を伝えます。ＣＰＵが割り込みを受けつけると割り込みコントローラは割り込みソースに応じて分岐先アドレスを発生する、という機能もあります。

ＰＣ－９８００シリーズでは割り込みによって、ＩＮＴ　８Ｈ～ＩＮＴ　１６Ｈの命令が実行され各処理ルーチンへ分岐します。

⊙ タイマ

μＰＤ８２５３ＣというタイマＬＳＩが使われています。３つのカウンタを内蔵していますが、インターバルタイマとしてはそのうちのカウンタ０を使用します。インターバルタイマの原理は基となるクロックをセットしたカウンタ値で分周するもので、セットしたカウンタ値に比例した一定時間ごとに割り込みを発生します。

基となるクロックは８ＭＨｚモードで1.9968ＭＨｚ（500.8ｎｓ）、５ＭＨｚモードで2.4576ＭＨｚ（406.9ｎｓ）です。これからカウンタ値はいくつに

したらよいかわかります。例えば、100μSECのインターバルタイマが欲しければ、８ＭＨＺなら200、５ＭＨＺなら246にすればよいでしょう。

インターバルタイマによる割り込みが発生するとＩＮＴ　８Ｈが実行されますから、ＩＮＴ　８Ｈに相当するベクターテーブルを書き換えてユーザの処理ルーチンを指すようにしておくことによって、インターバルタイマを使うことができます。本書に掲載したプログラムのうち時間の測定をしているものはすべてこの方法を用いています。参考にしてください。

○別売ディスクサービスのお知らせ○

知的アドベンチャーゲームにチャレンジしませんか!!

プロテクトは、マニアにとっての「知的アドベンチャーゲーム」と呼ぶにふさわしい十分な魅力をもったものです。強力なプロテクトに挑戦することは自分のソフト、ハードの技術の高さを証明するものです。

しかし、プロテクトというものを紙面だけで解説するのには限界があります。そこで、別売ディスクサービス版として、本書で扱ったプロテクトおよびその応用を含め、100種類のプロテクトを施したディスケットを用意しました。

別売ディスクサービス版には、これら100種類のプロテクトのすべてをチェックするプログラムが含まれております。このディスケットで立ち上げることによって、プログラムが起動するようになっています。読者の皆さんは、このディスケットの、できるだけ忠実な複製を作るよう挑戦してください。作成した複製をドライブ装置に入れて立ち上げれば、あなたの複製が100種類のうち、いくつまでクリアしたかを採点して表示します。

また、別売ディスクサービス版にはディスケットに組み込まれたプロテクトとは別に、本書に掲載した全プログラムのソース、オブジェクトを収録してあります。打ち込む手間が省略できます。ご利用ください。なお、これらのファイルにはまったくプロテクトはかかっておりませんので、ご安心ください。"xfiles.n88"（N_{88}－ＢＡＳＩＣのファイルの場合）、"PIP"（ＣＰ／Ｍ－８６の場合）、"COPY"（ＭＳ－ＤＯＳの場合）などで自由にコピーしていただけます。

3.5 インチ２ＤＤ版	定価７００４円	
5 インチ２ＤＤ版	定価５９７４円	
5 インチ２ＨＤ版	定価７００４円	
8 インチ２Ｄ 版	定価７００４円	送料各４００円

お求めは書店、販売店、または直接弊社宛てに製品名、住所、電話番号等を明記の上、下記送付先まで送料（３００円）を添えてお申し込みください。

秀和システムトレーディング株式会社サービスセンター
〒１０７　東京都港区赤坂8-5-29　チューカービル

ＰＣ―9800シリーズ
ザ・プロテクト

ISBN-87966-049-3
C3055 ¥1900E

○

発行日　1985年11月３日　第１刷
　　　　1989年２月10日　第28刷

○

著　者　　井上　智博

○

発行者　　牧谷　秀昭

○

編　集　　斉藤　和邦

○

発 行 所　　秀和システムトレーディング株式会社
郵便番号107　東京都港区赤坂8-5-29　チェッカービル

SHUWA SYSTEM TRADING CO.,LTD.
CHECKER BLDG.
8-5-29 AKASAKA, MINATO-KU TOKYO, 107 JAPAN

○

印刷所　　共同印刷株式会社

○

落丁、乱丁本はお取り替え致します。
なお、本書に関するご質問は小社編集部宛て、
ご質問の主旨、住所、氏名、電話番号等明記の上
書面にてお願い致します。

PC-9800シリーズ
ザ・プロテクト

発行日　1985・11・3　初版第1刷発行
　　　　1989・8・8　初版第31刷発行

著　者　井上智博

発行者　牧谷秀昭

発行所　秀和システムトレーディング株式会社
東京都港区赤坂8-5-29チェッカービル 〒107
03-470-4941

印刷所　共同印刷株式会社

ISBN4-87966-049-3 C3055

秀和システムトレーディング株式会社　定価1,957円(本体1,900円)

ISBN4-87966-049-3 C3055 P1957E